GENERALIZING MONOCULAR 3D OBJECT DETECTION

By

Abhinav Kumar

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2025

ABSTRACT

Monocular 3D object detection (Mono3D) is a fundamental computer vision task that estimates an object's class, 3D position, dimensions, and orientation from a single image. Its applications, including autonomous driving, augmented reality, and robotics, critically rely on accurate 3D environmental understanding. This thesis addresses the challenge of generalizing Mono3D models to diverse scenarios, including occlusions, datasets, object sizes, and camera parameters. To enhance occlusion robustness, we propose a mathematically differentiable NMS (GrooMeD-NMS). To improve generalization to new datasets, we explore depth equivariant (DEVIANT) backbones. We address the issue of large object detection, demonstrating that it's not solely a data imbalance or receptive field problem but also a noise sensitivity issue. To mitigate this, we introduce a segmentation-based approach in bird's-eye view with dice loss (SeaBird). Finally, we mathematically analyze the extrapolation of Mono3D models to unseen camera heights and improve Mono3D generalization in such out-of-distribution settings.

Dedicated to my country India.

ACKNOWLEDGEMENTS

My long PhD journey is the result of all my advisors, mentors, collaborators, friends, and family.

First and foremost, I express my deepest gratitude to my advisor, Prof. Xiaoming Liu. Prof. Liu took a bet on me at a point when I was lost in the dark - having an intense desire to do the PhD, but bereft of any support. Over the years, his taste, rigor, work-ethics and guidance has instilled in me an awareness of what it takes to do great research. The faith he had on my capabilities, even when I did not have on myself, is what I am grateful to him for.

I also thank my PhD committee – Prof. Daniel Morris (MSU), Prof. Georgia Gkioxari (Caltech, FAIR), Prof. Vishnu Boddetti (MSU), and Prof. Yu Kong (MSU) for agreeing to serve in my committee and supporting this journey. I acknowledge Prof. Daniel Morris for a three-year long collaboration on the Radar-Camera project, and sharing all his insights in developing radar-camera 3D detectors. I thank Prof. Georgia Gkioxari, who was also my internship manager at FAIR, Meta AI. Her vision expanded my horizons by giving me a taste of moonshot industry grade research, and what it takes to do one. I thank Prof. Vishnu Bodetti and Prof. Yu Kong for asking thought-provoking questions in this journey.

I deeply acknowledge my mentors: Dr. Tim Marks, Dr. Michael Jones, Dr. Anoop Cherian, Dr. Ye Wang, Dr. Toshi Koike-Akino and Prof. Cheng Feng at MERL. They took a bet on me as a first year PhD student when I didn't have any significant publications. The work done there culminated into my first CVPR paper. The paper opened doors to MSU to continue my PhD. If not for that internship, my aspirations for a PhD would have come to a crashing end five years back.

When I joined MSU, Dr. Garrick Brazil took me under his wings for a very daunting area of 3D computer vision, and was almost a second advisor to me at MSU and FAIR. It was due to his strong belief in me that I applied to FAIR internship, which at that time, I believed was beyond my capacity.

I acknowledge Dr. Yuliang Guo, Dr. Xinyu Huang and Dr. Liu Ren from Bosch AI Research. We had a long collaboration that spanned across 1 internship and 2 CVPR submissions. Their continued guidance and support enabled us to tackle hard open problems.

Dr. SriGanesh Madhvanath was my manager at Xerox Research, Bangalore. His mentorship gave me an early realization that as much as the calibre of a candidate matters, the environment and support system matters too. His generous endorsement opened doors for doing PhD in the US. As I grow in my career, I hope to pay it forward. I also thank Vladimir Kozitsky for his outstanding leadership on the LPRv2 project at Xerox Research.

Throughout my PhD journey in the US, I have been told that my Maths and Linear Algebra skills are decent. My Master's advisor, Prof. Animesh Kumar at IIT Bombay, is the stalwart who should get due credit. I stand on his shoulders and am deeply grateful to him for a rigorous foundation in mathematical thinking.

I also thank my professors at IIT Patna: Prof. Ayash Kanto Mukherjee, Prof. Kailash Ray, Prof. Lokman Hakim Choudhury, Prof. Nutan Tomar, Prof. Somnath Sarangi, Prof. Sumanta Gupta, and Prof. Yatendra Singh for their rigorous undergrad training. I thank my Physics and Maths +2 teacher Jitendra Bharadwaj for his inspirational and thought-provoking teaching, and my teachers at MKDAV Public School, Daltonganj: Antariksh Roy, Asha Mishra, Ashok Verma, Ganga Agarwal, Kunal Kumar, and Rita Sinha for laying a solid foundation to my higher studies.

Additionally, I thank Vincent Mattison and Brenda Hodge, the program coordinator and secretary in the CSE department at MSU for helping me with admin issues every single time.

This research would not have been possible without the funding from Ford Motor Company and Bosch AI Research. I gratefully acknowledge their financial support.

Prof. Liu's lab gave me an open culture, access to like-minded peers, and exposure to a setup for doing high quality research. I am thankful to all these amazing people in the lab: Prof. Feng Liu, Dr. Amin Jourabloo, Dr. Xi Yin, Dr. Garrick Brazil, Dr. Yaojie Liu, Shengjie Zhu, Andrew Hou, Vishal Asnani, Masa Hu, Yunfei Long, Xiao Guo, Minchul Kim, Yiyang Su, Jei Zhu and Zhiyuan Ren for reviewing my ideas, critiquing my papers and open discussions. Also, the newer members of the group: Girish Ganeshan, Dinqiang Ye, Zhihao Zhong, Zhizhong Huang, Hoang Le and Ziang Gu for sharing this journey with me. I am pretty sure each one of you have done and will be doing great in the future.

Next, I thank my friends in East Lansing - Bharat Basti Shenoy, Ankit Gupta, Rahul Dey, Ankit Kumar, Vishal Asnani, Hitesh Gakhar, Sachit Gaudi, Avrajit Ghosh and Ritam Guha, who made me feel East Lansing a second home.

I am grateful to my friends Koushik Chattopadhyay, Saurabh Kumar, Ashay Jain, Manas Pratim Haloi, Vidit Singh, and Priyanka Sinha for being my loudest supporters despite staying thousands of kilometers away. All of them have been friends for more than eight years with three for more than fifteen years. These were the people with whom I discussed all my PhD quitting plans.

I am also thankful to my parents, and my sister, Ayushi Raj, for their love, patience, support and encouragement, and keeping me sane during this demanding PhD journey.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Thesis Contributions 3
1.2 Thesis Organization 5

CHAPTER 2 GROOMED-NMS: GROUPED MATHEMATICALLY DIFFERENTIABLE NMS FOR MONOCULAR 3D OBJECT DETECTION 6
2.1 Introduction 6
2.2 Related Works 8
2.3 Background 10
2.4 GrooMeD-NMS 11
2.5 Experiments 18
2.6 Conclusions 24

CHAPTER 3 DEVIANT: DEPTH EQUIVARIANT NETWORK FOR MONOCULAR 3D OBJECT DETECTION 25
3.1 Introduction 25
3.2 Related Works 28
3.3 Background 29
3.4 Depth Equivariant Backbone 31
3.5 Experiments 34
3.6 Conclusions 42

CHAPTER 4 SEABIRD: SEGMENTATION IN BIRD'S VIEW WITH DICE LOSS IMPROVES MONOCULAR 3D DETECTION OF LARGE OBJECTS . 44
4.1 Introduction 44
4.2 Related Works 47
4.3 SeaBird 49
4.4 Experiments 53
4.5 Conclusions 60

CHAPTER 5 CHARM3R: TOWARDS CAMERA HEIGHT AGNOSTIC MONOCULAR 3D OBJECT DETECTOR 61
5.1 Introduction 61
5.2 Related Works 64
5.3 Notations and Preliminaries 66
5.4 CHARM3R 68
5.5 Experiments 73
5.6 Conclusions 77

CHAPTER 6 CONCLUSIONS AND FUTURE RESEARCH 78

BIBLIOGRAPHY 80

APPENDIX A PUBLICATIONS 103

APPENDIX B GROOMED-NMS APPENDIX 105
APPENDIX C DEVIANT APPENDIX 114
APPENDIX D SEABIRD APPENDIX 138
APPENDIX E CHARM3R APPENDIX 159

CHAPTER 1

INTRODUCTION

Monocular 3D object detection (Mono3D) is a fundamental computer vision problem that estimates an object's 3D position, dimensions, and orientation in a scene from a single image and its camera matrix. Its applications, including autonomous driving [108, 132, 181], robotics [213], and augmented reality [2, 172, 183, 293], critically rely on accurate 3D environmental understanding. To address these applications' demands, Mono3D networks must generalize across occlusions, diverse datasets [108], object sizes [110], camera intrinsics [14], extrinsics [94, 104], rotations [177], weather and geographical conditions [54] and be robust to adversarial examples [310].

Although Mono3D popularity stems from its high accessibility from consumer vehicles compared to LiDAR/Radar-based detectors [155, 215, 290] and computational efficiency compared to stereo-based detectors [34], Mono3D methods suffer from classical scale-depth ambiguity making their generalization harder. This is why there are fewer works along the lines of generalizing Mono3D. This thesis aims to generalize Mono3D to these varying conditions.

Most Mono3D networks benefit from end-to-end learning idea. However, they train without including NMS in the training pipeline making the final box after NMS outside the training paradigm. While there were attempts to include NMS in the training pipeline for tasks such as 2D object detection, they have been less widely adopted due to a non-mathematical expression of the NMS. We present and integrate GrooMeD-NMS– a novel Grouped Mathematically Differentiable NMS for Mono3D, such that the network is trained end-to-end with a loss on the boxes after NMS. We first formulate NMS as a matrix operation and then group and mask the boxes in an unsupervised manner to obtain a simple closed-form expression of the NMS. GrooMeD-NMS addresses the mismatch between training and inference pipelines and, therefore, forces the network to select the best 3D box in a differentiable manner. As a result, GrooMeD-NMS achieves state-of-the-art monocular 3D object detection results on the KITTI benchmark dataset performing comparably to monocular video-based methods, and outperforming them on the hard occluded examples.

Generalizing to datasets requires features which are dataset-independent. One common way is to obtain such features is incorporating inductive bias or symmetries in the network. One such symmetry is translating the ego camera along depth should result in deterministic transformations of the feature maps. Modern neural networks use building blocks such as convolutions that are equivariant to arbitrary 2D translations in the Euclidean manifold. However, these vanilla blocks are not equivariant to arbitrary 3D translations in the projective manifold. Even then, all Mono3D networks use vanilla blocks to obtain the 3D coordinates, a task for which the vanilla blocks are not designed for. This paper takes the first step towards convolutions equivariant to arbitrary 3D translations in the projective manifold. Since the depth is the hardest to estimate for monocular detection, this paper proposes Depth Equivariant Network (DEVIANT) built with existing scale equivariant steerable blocks. As a result, DEVIANT is equivariant to the depth translations in the projective manifold whereas vanilla networks are not. The additional depth equivariance forces the DEVIANT to learn consistent depth estimates, and therefore, DEVIANT works better than vanilla networks in cross-dataset evaluation. DEVIANT also achieves state-of-the-art monocular 3D detection results on KITTI and Waymo datasets in the image-only category and performs competitively to methods using extra information.

Mono3D networks achieve remarkable performance on cars and smaller objects. However, their performance drops on larger objects, leading to fatal accidents. Large objects like trailers, buses and trucks are harder to detect [268] in Mono3D, sometimes resulting in fatal accidents [23, 60]. Some attribute these failures to training data scarcity [308] or the receptive field requirements [268] of large objects. We find that modern frontal detectors struggle to generalize to large objects even on nearly balanced datasets. We argue that the cause of failure is the sensitivity of depth regression losses to noise of larger objects. To bridge this gap, we comprehensively investigate regression and dice losses, examining their robustness under varying error levels and object sizes. We mathematically prove that the dice loss leads to superior noise-robustness and model convergence for large objects compared to regression losses for a simplified case. Leveraging our theoretical insights, we propose SeaBird (Segmentation in Bird's View) as the first step towards generalizing to large

objects. SeaBird effectively integrates BEV segmentation on foreground objects for 3D detection, with the segmentation head trained with the dice loss. SeaBird achieves SoTA results on the KITTI-360 leaderboard and improves existing detectors on the nuScenes leaderboard, particularly for large objects.

With all these generalizations, the networks do not generalize well to changing extrinsics or viewpoints in testing. Finally, we aim to extend Mono3D's capabilities to varying camera extrinsics, such as camera heights.

1.1 Thesis Contributions

The thesis focuses on generalizing Mono3D across occlusions, datasets, object sizes, and camera extrinsics. The scale-depth ambiguity in Mono3D task requires elegant handling of the depth error.

- This thesis introduces the mathematically differentiable Non-Maximal Suppression, which attempts Mono3D generalization to occluded and hard objects. Most detectors use a post-processing algorithm called Non-Maximal Suppression (NMS) only during inference. While there were attempts to include NMS in the training pipeline for tasks such as 2D object detection, they have been less widely adopted due to a non-mathematical expression of the NMS. In this chapter, we present and integrate GrooMeD-NMS – a novel Grouped Mathematically Differentiable NMS for monocular 3D object detection, such that the network is trained end-to-end with a loss on the boxes after NMS. We first formulate NMS as a matrix operation and then group and mask the boxes in an unsupervised manner to obtain a simple closed-form expression of the NMS. GrooMeD-NMS addresses the mismatch between training and inference pipelines and, therefore, forces the network to select the best 3D box in a differentiable manner. As a result, GrooMeD-NMS achieves state-of-the-art monocular 3D object detection results on the KITTI dataset.
- We next propose the depth equivariant backbone in the projective manifold which attempts generalization to unseen datasets. Modern neural networks use building blocks such as convolutions that are equivariant to arbitrary 2D translations in the Euclidean manifold. However, these vanilla blocks are not equivariant to arbitrary 3D translations in the projective manifold.

Even then, all monocular 3D detectors use vanilla blocks to obtain the 3D coordinates, a task for which the vanilla blocks are not designed for. This chapter takes the first step towards convolutions equivariant to arbitrary 3D translations in the projective manifold. Since the depth is the hardest to estimate for monocular detection, this chapter proposes Depth Equivariant Network (DEVIANT) built with existing scale equivariant steerable blocks. As a result, DEVIANT is equivariant to the depth translations in the projective manifold whereas vanilla networks are not. The additional depth equivariance forces the DEVIANT to learn consistent depth estimates, and therefore, DEVIANT achieves state-of-the-art monocular 3D detection results on KITTI and Waymo datasets in the image-only category and performs competitively to methods using extra information. Moreover, DEVIANT works better than vanilla networks in cross-dataset evaluation.

- We then investigate large object detection, demonstrating that it is not solely a data imbalance or receptive field issue but also a noise sensitivity problem. To generalize Mono3D to large objects, it introduces a segmentation-based approach in bird's eye view with dice loss (SeaBird). Monocular 3D detectors achieve remarkable performance on cars and smaller objects. However, their performance drops on larger objects, leading to fatal accidents. Some attribute the failures to training data scarcity or their receptive field requirements of large objects. In this chapter, we highlight this understudied problem of generalization to large objects. We find that modern frontal detectors struggle to generalize to large objects even on nearly balanced datasets. We argue that the cause of failure is the sensitivity of depth regression losses to noise of larger objects. To bridge this gap, we comprehensively investigate regression and dice losses, examining their robustness under varying error levels and object sizes. We mathematically prove that the dice loss leads to superior noise-robustness and model convergence for large objects compared to regression losses for a simplified case. Leveraging our theoretical insights, we propose SeaBird (Segmentation in Bird's View) as the first step towards generalizing to large objects. SeaBird effectively integrates BEV segmentation on foreground objects for 3D detection, with the segmentation head trained with the dice loss. SeaBird achieves SoTA results on the KITTI-

360 leaderboard and improves existing detectors on the nuScenes leaderboard, particularly for large objects.

- Monocular 3D object detectors, while effective on data from one ego camera height, struggle with unseen or out-of-distribution camera heights. Existing methods often rely on Plucker embeddings, image transformations or data augmentation. This chapter takes a step towards this understudied problem by investigating the impact of camera height variations on state-of-the-art (SoTA) Mono3D models. With a systematic analysis on the extended CARLA dataset with multiple camera heights, we observe that depth estimation is a primary factor influencing performance under height variations. We mathematically prove and also empirically observe consistent negative and positive trends in mean depth error of regressed and ground-based depth models, respectively, under camera height changes. To mitigate this, we propose Camera Height Robust Monocular 3D Detector (CHARM3R), which averages both depth estimates within the model. CHARM3R significantly improves generalization to unseen camera heights, achieving SoTA performance on the CARLA dataset.

1.2 Thesis Organization

We organize the remaining chapters of the dissertation as follows. Chapter 2 introduces the mathematically differentiable Non-Maximal Suppression, which attempts generalization to occluded and hard objects. Chapter 3 describes the depth equivariant backbone which attempts generalization to unseen datasets. Chapter 4 investigates large object detection, demonstrating that it is not solely a data imbalance or receptive field issue but also a noise sensitivity problem. To improve large object detection, it introduces a segmentation-based approach in bird's eye view with dice loss (SeaBird). Chapter 5 attempts solving the generalization of Mono3D trained on single camera height to multiple camera heights. Chapter 6 introduces the future research for monocular 3D detection.

CHAPTER 2

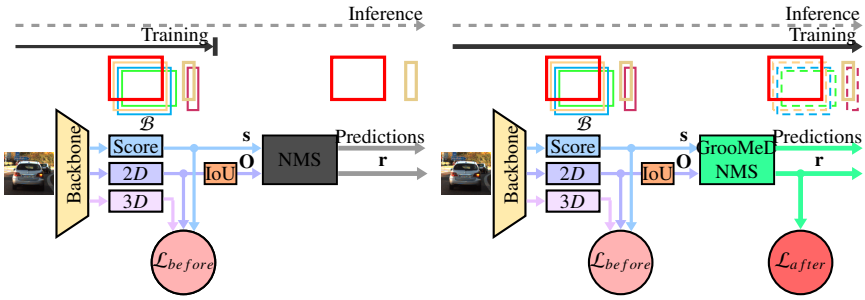
GROOMED-NMS: GROUPED MATHEMATICALLY DIFFERENTIABLE NMS FOR MONOCULAR 3D OBJECT DETECTION

Modern 3D object detectors have immensely benefited from the end-to-end learning idea. However, most of them use a post-processing algorithm called Non-Maximal Suppression (NMS) only during inference. While there were attempts to include NMS in the training pipeline for tasks such as 2D object detection, they have been less widely adopted due to a non-mathematical expression of the NMS. In this chapter, we present and integrate GrooMeD-NMS– a novel Grouped Mathematically Differentiable NMS for monocular 3D object detection, such that the network is trained end-to-end with a loss on the boxes after NMS. We first formulate NMS as a matrix operation and then group and mask the boxes in an unsupervised manner to obtain a simple closed-form expression of the NMS. GrooMeD-NMS addresses the mismatch between training and inference pipelines and, therefore, forces the network to select the best 3D box in a differentiable manner. As a result, GrooMeD-NMS achieves state-of-the-art monocular 3D object detection results on the KITTI benchmark dataset performing comparably to monocular video-based methods.

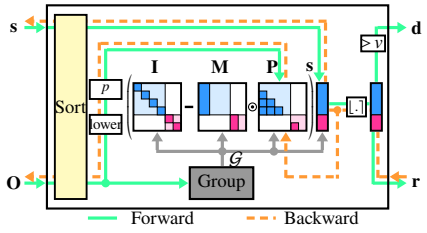
2.1 Introduction

3D object detection is one of the fundamental problems in computer vision, where the task is to infer 3D information of the object. Its applications include augmented reality [2, 201], robotics [120, 213], medical surgery [203], and, more recently path planning and scene understanding in autonomous driving [33, 90, 123, 220]. Most of the 3D object detectors [33, 90, 121, 123, 220] are extensions of the 2D object detector Faster R-CNN [202], which relies on the end-to-end learning idea to achieve State-of-the-Art (SoTA) object detection. Some of these methods have proposed changing architectures [123, 216, 220] or losses [15, 35]. Others have tried incorporating confidence [17, 216, 220] or temporal cues [17].

Almost all of them output a massive number of boxes for each object and, thus, rely on post-processing with a greedy [192] clustering algorithm called Non-Maximal Suppression (NMS) during inference to reduce the number of false positives and increase performance. However,



(a) Conventional NMS Pipeline (b) GrooMeD-NMS Pipeline



(c) GrooMeD-NMS layer

Figure 2.1 **Overview of GrooMeD-NMS.** (a) Conventional object detection has a mismatch between training and inference as it uses NMS only in inference. (b) To address this, we propose a novel GrooMeD-NMS layer, such that the network is trained end-to-end with NMS applied. **s** and **r** denote the score of boxes $\mathcal{B}$ before and after the NMS respectively. **O** denotes the matrix containing IoU_{2D} overlaps of $\mathcal{B}$. $\mathcal{L}_{before}$ denotes the losses before the NMS, while $\mathcal{L}_{after}$ denotes the loss after the NMS. (c) GrooMeD-NMS layer calculates **r** in a differentiable manner giving gradients from $\mathcal{L}_{after}$ when the best-localized box corresponding to an object is not selected after NMS.

these works have largely overlooked NMS's inclusion in training leading to an apparent *mismatch* between training and inference pipelines as the losses are applied on all boxes before NMS but not on *final* boxes after NMS (see Fig. 2.1a). We also find that 3D object detection suffers a greater mismatch between classification and 3D localization compared to that of 2D localization, as discussed further in Sec. B.3.2 of the supplementary and observed in [17, 90, 216]. Hence, our focus is 3D object detection.

Earlier attempts to include NMS in the training pipeline [80, 81, 192] have been made for 2D object detection where the improvements are less visible. Recent efforts to improve the correlation in 3D object detection involve calculating [220, 222] or predicting [17, 216] the scores via likelihood estimation [111] or enforcing the correlation explicitly [90]. Although this improves the 3D detection performance, improvements are limited as their training pipeline is not end to end

in the absence of a differentiable NMS.

To address the mismatch between training and inference pipelines as well as the mismatch between classification and 3D localization, we propose including the NMS in the training pipeline, which gives a useful gradient to the network so that it figures out which boxes are the best-localized in 3D and, therefore, should be ranked higher (see Fig. 2.1b).

An ideal NMS for inclusion in the training pipeline should be not only differentiable but also parallelizable. Unfortunately, the inference-based classical NMS and Soft-NMS [12] are greedy, set-based and, therefore, not parallelizable [192]. To make the NMS parallelizable, we first formulate the classical NMS as matrix operation and then obtain a closed-form mathematical expression using elementary matrix operations such as matrix multiplication, matrix inversion, and clipping. We then replace the threshold pruning in the classical NMS with its softer version [12] to get useful gradients. These two changes make the NMS GPU-friendly, and the gradients are backpropagated. We next group and mask the boxes in an unsupervised manner, which removes the matrix inversion and simplifies our proposed differentiable NMS expression further. We call this NMS as Grouped Mathematically Differentiable NMS (GrooMeD-NMS).

In summary, the main contributions of this work include:

- This is the first work to propose and integrate a closed-form mathematically differentiable NMS for object detection, such that the network is trained end-to-end with a loss on the boxes after NMS.
- We propose an unsupervised grouping and masking on the boxes to remove the matrix inversion in the closed-form NMS expression.
- We achieve SoTA monocular 3D object detection performance on the KITTI dataset performing comparably to monocular video-based methods.

2.2 Related Works

3D Object Detection. Recent success in 2D object detection [69, 70, 139, 200, 202] has inspired people to infer 3D information from a single 2D (monocular) image. However, the monocular problem is ill-posed due to the inherent scale/depth ambiguity [232]. Hence, approaches use

additional sensors such as LiDAR [90, 215, 269], stereo [122, 254] or radar [178, 242]. Although LiDAR depth estimations are accurate, LiDAR data is sparse [85] and computationally expensive to process [232]. Moreover, LiDAR s are expensive and do not work well in severe weather [232].

Hence, there have been several works on monocular 3D object detection. Earlier approaches [31, 61,186,187] use hand-crafted features, while the recent ones are all based on deep learning. Some of these methods have proposed changing architectures [123, 143, 232] or losses [15, 35]. Others have tried incorporating confidence [17,143,216,220], augmentation [223], depth in convolution [15,52] or temporal cues [17]. Our work proposes to incorporate NMS in the training pipeline of monocular 3D object detection.

Non-Maximal Suppression. NMS has been used to reduce false positives in edge detection [206], feature point detection [75, 157, 174], face detection [243], human detection [16, 18, 47] as well as SoTA 2D [69, 139, 200, 202] and 3D detection [5, 17, 33, 216, 220, 232]. Modifications to NMS in 2D detection [12, 49, 80, 81, 192], 2D pedestrian detection [116, 145, 209], 2D salient object detection [298] and 3D detection [216] can be classified into three categories – inference NMS [12, 216], optimization-based NMS [3, 49, 116, 209, 244, 298] and neural network based NMS [78, 80, 81, 145, 192].

The inference NMS [12] changes the way the boxes are pruned in the final set of predictions. [216] uses weighted averaging to update the z-coordinate after NMS. [209] solves quadratic unconstrained binary optimization while [3,116,224] and [298] use point processes and MAP based inference respectively. [49] and [244] formulate NMS as a structured prediction task for isolated and all object instances respectively. The neural network NMS use a multi-layer network and message-passing to approximate NMS [80, 81, 192] or to predict the NMS threshold adaptively [145]. [78] approximates the sub-gradients of the network without modelling NMS via a transitive relationship. Our work proposes a grouped closed-form mathematical approximation of the classical NMS and does not require multiple layers or message-passing. We detail these differences in Sec. 2.4.2.

2.3 Background

2.3.1 Notations

Let $\mathcal{B} = \{b_i\}_{i=1}^n$ denote the set of boxes or proposals b_i from an image. Let $\mathbf{s} = \{s_i\}_{i=1}^n$ and $\mathbf{r} = \{r_i\}_{i=1}^n$ denote their scores (before NMS) and rescores (updated scores after NMS) respectively such that $r_i, s_i \geq 0 \; \forall \; i$. $\mathcal{D}$ denotes the subset of $\mathcal{B}$ after the NMS. Let $\mathbf{O} = [o_{ij}]$ denote the $n \times n$ matrix with o_{ij} denoting the 2D Intersection over Union (IoU_{2D}) of b_i and b_j. The *pruning* function p decides how to rescore a set of boxes $\mathcal{B}$ based on IoU_{2D} overlaps of its neighbors, sometimes suppressing boxes entirely. In other words, $p(o_i) = 1$ denotes the box b_i is suppressed while $p(o_i) = 0$ denotes b_i is kept in $\mathcal{D}$. The NMS threshold N_t is the threshold for which two boxes need in order for the non-maximum to be suppressed. The temperature τ controls the shape of the exponential and sigmoidal pruning functions p. v thresholds the rescores in GrooMeD and Soft-NMS [13] to decide if the box remains valid after NMS.

$\mathcal{B}$ is partitioned into different groups $\mathcal{G} = \{\mathcal{G}_k\}$. $\mathcal{B}_{\mathcal{G}_k}$ denotes the subset of $\mathcal{B}$ belonging to group k. Thus, $\mathcal{B}_{\mathcal{G}_k} = \{b_i\} \; \forall \; b_i \in \mathcal{G}_k$ and $\mathcal{B}_{\mathcal{G}_k} \cap \mathcal{B}_{\mathcal{G}_l} = \phi \; \forall \; k \neq l$. $\mathcal{G}_k$ in the subscript of a variable denotes its subset corresponding to $\mathcal{B}_{\mathcal{G}_k}$. Thus, $\mathbf{s}_{\mathcal{G}_k}$ and $\mathbf{r}_{\mathcal{G}_k}$ denote the scores and the rescores of $\mathcal{B}_{\mathcal{G}_k}$ respectively. α denotes the maximum group size.

$\vee$ denotes the logical OR while $\lfloor x \rceil$ denotes clipping of x in the range $[0, 1]$. Formally,

$$\lfloor x \rceil = \begin{cases} 1, & x > 1 \\ x, & 0 \leq x \leq 1 \\ 0, & x < 0 \end{cases} \tag{2.1}$$

$|\mathbf{s}|$ denotes the number of elements in $\mathbf{s}$. $\llcorner$ in the subscript denotes the lower triangular version of the matrix without the principal diagonal. $\odot$ denotes the element-wise multiplication. $\mathbf{I}$ denotes the identity matrix.

2.3.2 Classical and Soft-NMS

NMS is one of the building blocks in object detection whose high-level goal is to iteratively suppress boxes which have too much IoU with a nearby high-scoring box. We first give an overview of the classical and Soft-NMS [12], which are greedy and used in inference. Classical NMS uses

Algorithm 1: Classical/Soft-NMS [12]

Input: $\mathbf{s}$: scores, $\mathbf{O}$: IoU_{2D} matrix, N_t: NMS threshold, p: pruning function, τ: temperature

Output: $\mathbf{d}$: box index after NMS, $\mathbf{r}$: scores after NMS

1 **begin**
2 $\quad \mathbf{d} \leftarrow \{\}$
3 $\quad \mathbf{t} \leftarrow \{1, \cdots, |\mathbf{s}|\}$ ▷ All box indices
4 $\quad \mathbf{r} \leftarrow \mathbf{s}$
5 $\quad$ **while** $\mathbf{t} \neq empty$ **do**
6 $\quad\quad \nu \leftarrow \text{argmax } \mathbf{r}[\mathbf{t}]$ ▷ Top scored box
7 $\quad\quad \mathbf{d} \leftarrow \mathbf{d} \cup \nu$ ▷ Add to valid box index
8 $\quad\quad \mathbf{t} \leftarrow \mathbf{t} - \nu$ ▷ Remove from $\mathbf{t}$
9 $\quad\quad$ **for** $i \leftarrow 1 : |\mathbf{t}|$ **do**
10 $\quad\quad\quad r_i \leftarrow (1 - p_\tau(\mathbf{O}[\nu, i]))r_i$ ▷ Rescore
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 **end**

the idea that the score of a box having a high IoU_{2D} overlap with *any* of the selected boxes should be suppressed to zero. That is, it uses a hard pruning p without any temperature τ. Soft-NMS makes this pruning soft via temperature τ. Thus, classical and Soft-NMS only differ in the choice of p. We reproduce them in Alg. 1 using our notations.

2.4 GrooMeD-NMS

Classical NMS (Alg. 1) uses argmax and greedily calculates the rescore r_i of boxes $\mathcal{B}$ and, is thus not parallelizable or differentiable [192]. We wish to find its smooth approximation in closed-form for including in the training pipeline.

2.4.1 Formulation

2.4.1.1 Sorting

Classical NMS uses the non-differentiable hard argmax operation (Line 6 of Alg. 1). We remove the argmax by hard sorting the scores $\mathbf{s}$ and $\mathbf{O}$ in decreasing order (lines 2-3 of Alg. 2). We also try making the sorting soft. Note that we require the permutation of $\mathbf{s}$ to sort $\mathbf{O}$. Most soft sorting methods [8, 10, 185, 190] apply the soft permutation to the same vector. Only two other methods [46, 191] can apply the soft permutation to another vector. Both methods use $O\left(n^2\right)$

Algorithm 2: GrooMeD-NMS

Input: $\mathbf{s}$: scores, $\mathbf{O}$: $\text{IoU}_{2\text{D}}$ matrix, N_t: NMS threshold, p: pruning function, v: valid box threshold, α: maximum group size

Output: $\mathbf{d}$: box index after NMS, $\mathbf{r}$: scores after NMS

begin
- $\mathbf{s}$, index $\leftarrow$ sort($\mathbf{s}$, descending= True) ▷ Sort $\mathbf{s}$
- $\mathbf{O} \leftarrow \mathbf{O}$[index][:, index] ▷ Sort $\mathbf{O}$
- $\mathbf{O}_{\llcorner} \leftarrow$ lower($\mathbf{O}$) ▷ Lower △ular matrix
- $\mathbf{P} \leftarrow p(\mathbf{O}_{\llcorner})$ ▷ Prune matrix
- $\mathbf{I} \leftarrow$ Identity($|\mathbf{s}|$) ▷ Identity matrix
- $\mathcal{G} \leftarrow$ group($\mathbf{O}, N_t, \alpha$) ▷ Group boxes $\mathcal{B}$
- **for** $k \leftarrow 1 : |\mathcal{G}|$ **do**
  - $\mathbf{M}_{\mathcal{G}_k} \leftarrow$ zeros $(|\mathcal{G}_k|, |\mathcal{G}_k|)$ ▷ Prepare mask
  - $\mathbf{M}_{\mathcal{G}_k}[:, \mathcal{G}_k[1]] \leftarrow 1$ ▷ First col of $\mathbf{M}_{\mathcal{G}_k}$
  - $\mathbf{r}_{\mathcal{G}_k} \leftarrow \left\lfloor \left(\mathbf{I}_{\mathcal{G}_k} - \mathbf{M}_{\mathcal{G}_k} \odot \mathbf{P}_{\mathcal{G}_k}\right) \mathbf{s}_{\mathcal{G}_k} \right\rceil$ ▷ Rescore
- **end**
- $\mathbf{d} \leftarrow$ index[$\mathbf{r}$ >= v] ▷ Valid box index

end

computations for soft sorting [10]. We implement [191] and find that [191] is overly dependent on temperature τ to break out the ranks, and its gradients are too unreliable to train our model. Hence, we stick with the hard sorting of $\mathbf{s}$ and $\mathbf{O}$.

2.4.1.2 NMS as a Matrix Operation

The rescoring process of the classical NMS is greedy set-based [192] and only considers overlaps with unsuppressed boxes. We first generalize this rescoring by accounting for the effect of all (suppressed and unsuppressed) boxes as

$$r_i \approx \max\left(s_i - \sum_{j=1}^{i-1} p(o_{ij}) r_j,\ 0\right) \tag{2.2}$$

using the relaxation of logical OR $\bigvee$ operator as $\sum$ [106, 124]. See Sec. B.1 of the supplementary material for an alternate explanation of Eq. (2.2). The presence of r_j on the RHS of Eq. (2.2) prevents suppressed boxes from influencing other boxes hugely. When p outputs discretely as $\{0, 1\}$ as in classical NMS, scores s_i are guaranteed to be suppressed to $r_i = 0$ or left unchanged

$r_i = s_i$ thereby implying $r_i \leq s_i \; \forall \; i$. We write the rescores $\mathbf{r}$ in a matrix formulation as

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} \approx \max \left(\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix} - \begin{bmatrix} 0 & 0 & \dots & 0 \\ p(o_{21}) & 0 & \dots & 0 \\ p(o_{31}) & p(o_{32}) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ p(o_{n1}) & p(o_{n2}) & \dots & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right). \tag{2.3}$$

The above equation is written compactly as

$$\mathbf{r} \approx \max(\mathbf{s} - \mathbf{P}\mathbf{r}, \mathbf{0}), \tag{2.4}$$

where $\mathbf{P}$, called the Prune Matrix, is obtained when the pruning function p operates element-wise on $\mathbf{O}_{\llcorner}$. Maximum operation makes Eq. (2.4) non-linear [112] and, thus, difficult to solve. However, to avoid recursion, we use

$$\mathbf{r} \approx \left\lfloor (\mathbf{I} + \mathbf{P})^{-1} \, \mathbf{s} \right\rceil, \tag{2.5}$$

as the solution to Eq. (2.4) with $\mathbf{I}$ being the identity matrix. Intuitively, if the matrix inversion is considered division in Eq. (2.5) and the boxes have overlaps, the rescores are the scores divided by a number greater than one and are, therefore, lesser than scores. If the boxes do not overlap, the division is by one and rescores equal scores.

Note that the $\mathbf{I} + \mathbf{P}$ in Eq. (2.5) is a lower triangular matrix with ones on the principal diagonal. Hence, $\mathbf{I} + \mathbf{P}$ is always full rank and, therefore, always invertible.

2.4.1.3 Grouping

We next observe that the object detectors output multiple boxes for an object, and a good detector outputs boxes wherever it finds objects in the monocular image. Thus, we cluster the boxes in an image in an unsupervised manner based on IoU_{2D} overlaps to obtain the groups $\mathcal{G}$. Grouping thus mimics the grouping of the classical NMS, but does not rescore the boxes. As clustering limits interactions to intra-group interactions among the boxes, we write Eq. (2.5) as

$$\mathbf{r}_{\mathcal{G}_k} \approx \left\lfloor \left(\mathbf{I}_{\mathcal{G}_k} + \mathbf{P}_{\mathcal{G}_k} \right)^{-1} \mathbf{s}_{\mathcal{G}_k} \right\rceil. \tag{2.6}$$

Algorithm 3: Grouping of boxes

Input: $\mathbf{O}$: sorted IoU_{2D} matrix, N_t: NMS threshold, α: maximum group size
Output: $\mathcal{G}$: Groups

begin
  $\mathcal{G} \leftarrow \{\}$
  $\mathbf{t} \leftarrow \{1, \cdots, \mathbf{O}.\text{shape}[1]\}$ ▷ All box indices
  while $\mathbf{t} \neq empty$ **do**
    $\mathbf{u} \leftarrow \mathbf{O}[:, 1] > N_t$ ▷ High overlap indices
    $\mathbf{v} \leftarrow \mathbf{t}[\mathbf{u}]$ ▷ New group
    $n_{\mathcal{G}_k} \leftarrow \min(|\mathbf{v}|, \alpha)$
    $\mathcal{G}.\text{insert}(\mathbf{v}[: n_{\mathcal{G}_k}])$ ▷ Insert new group
    $\mathbf{w} \leftarrow \mathbf{O}[:, 1] \leq N_t$ ▷ Low overlap indices
    $\mathbf{t} \leftarrow \mathbf{t}[\mathbf{w}]$ ▷ Keep $\mathbf{w}$ indices in $\mathbf{t}$
    $\mathbf{O} \leftarrow \mathbf{O}[\mathbf{w}][:, \mathbf{w}]$ ▷ Keep $\mathbf{w}$ indices in $\mathbf{O}$
  end
end

This results in taking smaller matrix inverses in Eq. (2.6) than Eq. (2.5).

We use a simplistic grouping algorithm, *i.e.*, we form a group $\mathcal{G}_k$ with boxes having high IoU_{2D} overlap with the top-ranked box, given that we sorted the scores. As the group size is limited by α, we choose a minimum of α and the number of boxes in $\mathcal{G}_k$. We next delete all the boxes of this group and iterate until we run out of boxes. Also, grouping uses IoU_{2D} since we can achieve meaningful clustering in 2D. We detail this unsupervised grouping in Alg. 3.

2.4.1.4 Masking

Classical NMS considers the IoU_{2D} of the top-scored box with other boxes. This consideration is equivalent to only keeping the column of $\mathbf{O}$ corresponding to the top box while assigning the rest of the columns to be zero. We implement this through masking of $\mathbf{P}_{\mathcal{G}_k}$. Let $\mathbf{M}_{\mathcal{G}_k}$ denote the binary mask corresponding to group $\mathcal{G}_k$. Then, entries in the binary matrix $\mathbf{M}_{\mathcal{G}_k}$ in the column corresponding to the top-scored box are 1 and the rest are 0. Hence, only one of the columns in $\mathbf{M}_{\mathcal{G}_k} \odot \mathbf{P}_{\mathcal{G}_k}$ is non-zero. Now, $\mathbf{I}_{\mathcal{G}_k} + \mathbf{M}_{\mathcal{G}_k} \odot \mathbf{P}_{\mathcal{G}_k}$ is a Frobenius matrix (Gaussian transformation) and we, therefore, invert this matrix by simply subtracting the second term [71]. In other words, $(\mathbf{I}_{\mathcal{G}_k} + \mathbf{M}_{\mathcal{G}_k} \odot \mathbf{P}_{\mathcal{G}_k})^{-1} = \mathbf{I}_{\mathcal{G}_k} - \mathbf{M}_{\mathcal{G}_k} \odot \mathbf{P}_{\mathcal{G}_k}$. Hence, we simplify Eq. (2.6) further to get

$$\mathbf{r}_{\mathcal{G}_k} \approx \left\lfloor \left(\mathbf{I}_{\mathcal{G}_k} - \mathbf{M}_{\mathcal{G}_k} \odot \mathbf{P}_{\mathcal{G}_k}\right) \mathbf{s}_{\mathcal{G}_k} \right\rceil. \tag{2.7}$$

Thus, masking allows to bypass the computationally expensive matrix inverse operation altogether.

We call the NMS based on Eq. (2.7) as Grouped Mathematically Differentiable Non-Maximal Suppression or GrooMeD-NMS. We summarize the complete GrooMeD-NMS in Alg. 2 and show its block-diagram in Fig. 2.1c. GrooMeD-NMS in Fig. 2.1c provides two gradients - one through $\mathbf{s}$ and other through $\mathbf{O}$.

2.4.1.5 Pruning Function

As explained in Sec. 2.3.1, the pruning function p decides whether to keep the box in the final set of predictions $\mathcal{D}$ or not based on IoU_{2D} overlaps, *i.e.*, $p(o_i) = 1$ denotes the box b_i is suppressed while $p(o_i) = 0$ denotes b_i is kept in $\mathcal{D}$.

Classical NMS uses the threshold as the pruning function, which does not give useful gradients. Therefore, we considered three different functions for p: Linear, a temperature (τ)-controlled Exponential, and Sigmoidal function.

- **Linear** Linear pruning function [12] is $p(o) = o$.
- **Exponential** Exponential pruning function [12] is $p(o) = 1 - \exp\left(-\frac{o^2}{\tau}\right)$.
- **Sigmoidal** Sigmoidal pruning function is $p(o) = \sigma\left(\frac{o-N_t}{\tau}\right)$ with σ denoting the standard sigmoid. Sigmoidal function appears as the binary cross entropy relaxation of the subset selection problem [185].

We show these pruning functions in Fig. 2.2. The ablation studies (Sec. 2.5.4) show that choosing p as Linear yields the simplest and the best GrooMeD-NMS.

2.4.2 Differences from Existing NMS

Although no differentiable NMS has been proposed for the monocular 3D object detection, we compare our GrooMeD-NMS with the NMS proposed for 2D object detection, 2D pedestrian detection, 2D salient object detection, and 3D object detection in Tab. 2.1. No method described in Tab. 2.1 has a matrix-based closed-form mathematical expression of the NMS. Classical, Soft [12] and Distance-NMS [216] are used at the inference time, while GrooMeD-NMS is used during both training and inference. Distance-NMS [216] updates the z-coordinate of the box after NMS as the weighted average of the z-coordinates of top-κ boxes. QUBO-NMS [209], Point-NMS [116, 224],

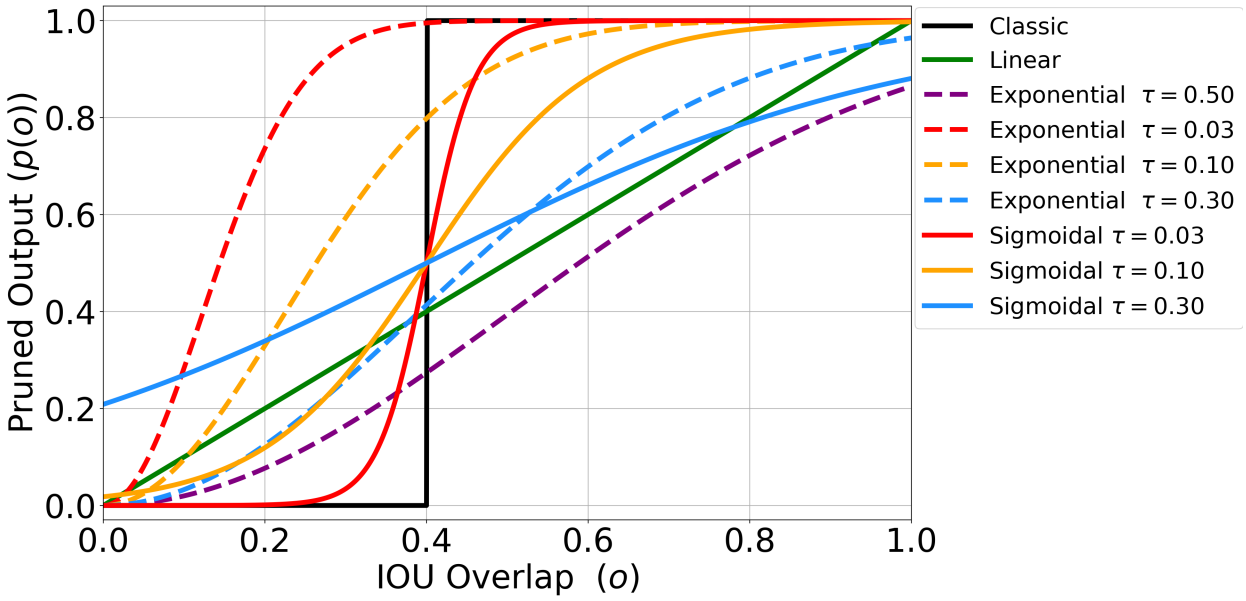


Figure 2.2 **Pruning functions** p of the classical and GrooMeD-NMS. We use the Linear and Exponential pruning of the Soft-NMS [12] while training with the GrooMeD-NMS.

and MAP-NMS [298] are not used in end-to-end training. [3] proposes a trainable Point-NMS. The Structured-SVM based NMS [49, 244] rely on structured SVM to obtain the rescores. Adaptive-NMS [145] uses a separate neural network to predict the classical NMS threshold N_t. The trainable neural network based NMS (NN-NMS) [80, 81, 192] use a separate neural network containing multiple layers and/or message-passing to approximate the NMS and do not use the pruning function. Unlike these methods, GrooMeD-NMS uses a single layer and does not require multiple layers or message passing. Our NMS is parallel up to group (denoted by $\mathcal{G}$). However, $|\mathcal{G}|$ is, in general, $<< |\mathcal{B}|$ in the NMS.

2.4.3 Target Assignment and Loss Function

Target Assignment. Our method consists of M3D-RPN [15] and uses binning and self-balancing confidence [17]. The boxes' self-balancing confidence are used as scores $\mathbf{s}$, which pass through the GrooMeD-NMS layer to obtain the rescores $\mathbf{r}$. The rescores signal the network if the *best* box has not been selected for a particular object.

We extend the notion of the best 2D box [192] to 3D. The best box has the highest product of IoU_{2D} and gIoU_{3D} [204] with ground truth g_l. If the product is greater than a certain threshold β,

Table 2.1 **Comparison of different NMS**. [Key: Train= End-to-end Trainable, Prune= Pruning function, #Layers= Number of layers, Par= Parallelizable]

NMS	Train	Rescore	Prune	#Layers	Par
Classical	×	×	Hard	-	$O(\|\mathcal{G}\|)$
Soft-NMS [12]	×	×	Soft	-	$O(\|\mathcal{G}\|)$
Distance-NMS [216]	×	×	Hard	-	$O(\|\mathcal{G}\|)$
QUBO-NMS [209]	×	Optimization	×	-	-
Point-NMS [116, 224]	×	Point Process	×	-	-
Trainable Point-NMS [3]	✓	Point Process	×	-	-
MAP-NMS [298]	×	MAP	×	-	-
Structured-NMS [49, 244]	×	SSVM	×	-	-
Adaptive-NMS [145]	×	×	Hard	>1	$O(\|\mathcal{G}\|)$
NN-NMS [80, 81, 192]	✓	Neural Network	×	>1	$O(1)$
GrooMeD-NMS (Ours)	✓	Matrix	Soft	1	$O(\|\mathcal{G}\|)$

it is assigned a positive label. Mathematically,

$$\text{target}(b_i) = \begin{cases} 1, & \text{if } \exists\ g_l \text{ st } i = \text{argmax}\, q(b_j, g_l) \\ & \qquad \text{and } q(b_i, g_l) \geq \beta \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

with $q(b_j, g_l) = \text{IoU}_{\text{2D}}(b_j, g_l)\ \left(\frac{1+\text{gIoU}_{\text{3D}}(b_j, g_l)}{2}\right)$. gIoU_{3D} is known to provide signal even for non-intersecting boxes [204], where the usual IoU_{3D} is always zero. Therefore, we use gIoU_{3D} instead of regular IoU_{3D} for figuring out the best box in 3D as many 3D boxes have a zero IoU_{3D} overlap with the ground truth. For calculating gIoU_{3D}, we first calculate the volume V and hull volume V_{hull} of the 3D boxes. V_{hull} is the product of gIoU_{2D} in Birds Eye View (BEV), removing the rotations and hull of the Y dimension. gIoU_{3D} is then given by

$$\text{gIoU}_{\text{3D}}(b_i, b_j) = \frac{V(b_i \cap b_j)}{V(b_i \cup b_j)} + \frac{V(b_i \cup b_j)}{V_{hull}(b_i, b_j)} - 1. \tag{2.9}$$

Loss Function. Generally the number of best boxes is less than the number of ground truths in an image, as there could be some ground truth boxes for which no box is predicted. The tiny number of best boxes introduces a far-heavier skew than the foreground-background classification. Thus, we use the modified AP-Loss [30] as our loss after NMS since AP-Loss does not suffer from class imbalance [30].

Vanilla AP-Loss treats boxes of all images in a mini-batch equally, and the gradients are back-propagated through all the boxes. We remove this condition and rank boxes in an image-wise manner. In other words, if the best boxes are correctly ranked in one image and are not in the second, then the gradients only affect the boxes of the second image. We call this modification of AP-Loss as *Imagewise* AP-Loss. In other words,

$$\mathcal{L}_{Imagewise} = \frac{1}{N}\sum_{m=1}^{N} \text{AP}(\mathbf{r}^{(m)}, \text{target}(\mathcal{B}^{(m)})), \tag{2.10}$$

where $\mathbf{r}^{(m)}$ and $\mathcal{B}^{(m)}$ denote the rescores and the boxes of the m^{th} image in a mini-batch respectively. This is different from previous NMS approaches [78, 80, 81, 192], which use classification losses. Our ablation studies (Sec. 2.5.4) show that the Imagewise AP-Loss is better suited to be used after NMS than the classification loss.

Our overall loss function is thus given by $\mathcal{L} = \mathcal{L}_{before} + \lambda\mathcal{L}_{after}$ where $\mathcal{L}_{before}$ denotes the losses before the NMS including classification, 2D and 3D regression as well as confidence losses, and $\mathcal{L}_{after}$ denotes the loss term after the NMS, which is the Imagewise AP-Loss with λ being the weight. See Sec. B.2 of the supplementary material for more details of the loss function.

2.5 Experiments

Our experiments use the most widely used KITTI autonomous driving dataset [67]. We modify the publicly-available PyTorch [184] code of Kinematic-3D [17]. [17] uses DenseNet-121 [86] trained on ImageNet as the backbone and $n_h = 1{,}024$ using 3D-RPN settings of [15]. As [17] is a video-based method while GrooMeD-NMS is an image-based method, we use the best image model of [17] henceforth called Kinematic (Image) as our baseline for a fair comparison. Kinematic (Image) is built on M3D-RPN [15] and uses binning and self-balancing confidence.

Data Splits. There are three commonly used data splits of the KITTI dataset; we evaluate our method on all three.

KITTI Test (Full) split: Official KITTI 3D benchmark [1] consists of 7,481 training and 7,518 testing images [67].

KITTI Val 1 split: It partitions the 7,481 training images into 3,712 training and 3,769 validation images [17, 32, 220].

Table 2.2 **KITTI Test cars** AP $_{3D|R_{40}}$ and AP $_{BEV|R_{40}}$ comparisons ($IoU_{3D} \geq 0.7$). Previous results are quoted from the official leader-board or from papers.[Key: **Best**, **Second Best**].

Method	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard
FQNet [143]	2.77	1.51	1.01	5.40	3.23	2.46
ROI-10D [169]	4.32	2.02	1.46	9.78	4.91	3.74
GS3D [121]	4.47	2.90	2.47	8.41	6.08	4.94
MonoGRNet [195]	9.61	5.74	4.25	18.19	11.17	8.73
MonoPSR [107]	10.76	7.25	5.85	18.33	12.58	9.91
MonoDIS [222]	10.37	7.94	6.40	17.23	13.19	11.12
UR3D [216]	15.58	8.61	6.00	21.85	12.51	9.20
M3D-RPN [15]	14.76	9.71	7.42	21.02	13.67	10.23
SMOKE [153]	14.03	9.76	7.84	20.83	14.49	12.75
MonoPair [35]	13.04	9.99	8.65	19.28	14.83	12.89
RTM3D [123]	14.41	10.34	8.77	19.17	14.20	11.99
AM3D [165]	16.50	10.74	**9.52**	25.03	17.32	**14.91**
MoVi-3D [223]	15.19	10.90	9.26	22.76	17.03	10.86
RAR-Net [144]	16.37	11.01	**9.52**	22.45	15.02	12.93
M3D-SSD [160]	17.51	11.46	8.98	24.15	15.93	12.11
DA-3Ddet [287]	16.77	11.50	8.93	-	-	-
D4LCN [52]	16.65	11.72	9.51	22.51	16.02	12.55
Kinematic (Video) [17]	**19.07**	**12.72**	9.17	**26.69**	**17.52**	13.10
GrooMeD-NMS (Ours)	**18.10**	**12.32**	**9.65**	**26.19**	**18.27**	**14.05**

KITTI Val 2 split: It partitions the 7,481 training images into 3,682 training and 3,799 validation images [271].

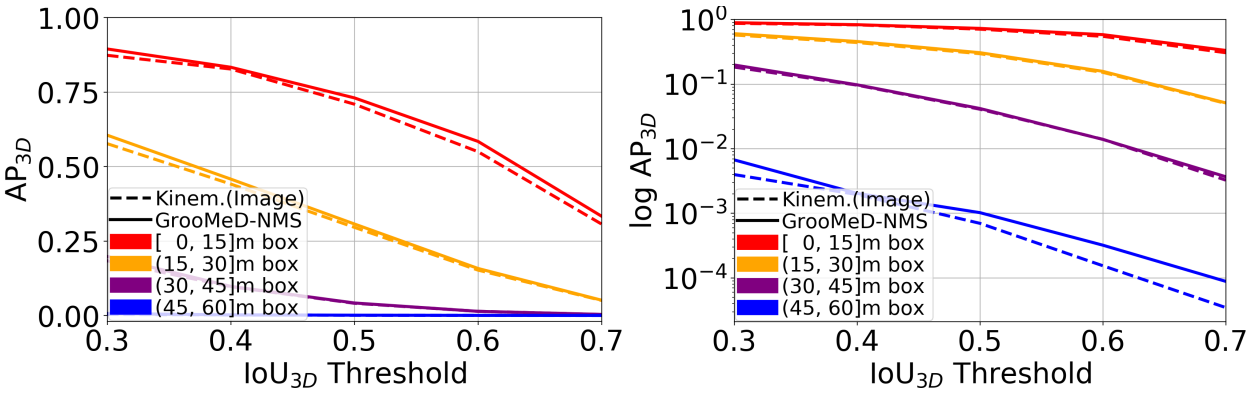
Training. Training is done in two phases - warmup and full [17]. We initialize the model with the confidence prediction branch from warmup weights and finetune using the self-balancing loss [17] and Imagewise AP-Loss [30] after our GrooMeD-NMS. See Sec. B.3.1 of the supplementary material for more training details. We keep the weight λ at 0.05. Unless otherwise stated, we use p as the Linear function (this does not require τ) with $\alpha = 100$. N_t, ν and β are set to 0.4 [15, 17], 0.3 and 0.3 respectively.

Inference. We multiply the class and predicted confidence to get the box's overall score in inference as in [99, 216, 241]. See Sec. 2.5.2 for training and inference times.

Evaluation Metrics. KITTI uses AP $_{3D|R_{40}}$ metric to evaluate object detection following [220, 222]. KITTI benchmark evaluates on three object categories: Easy, Moderate and Hard. It assigns each

Table 2.3 **KITTI Val 1 cars** AP $_{3D|R_{40}}$ and AP $_{BEV|R_{40}}$ results. [Key: **Best**, **Second Best**].

Method	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)			AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
MonoDR [7]	12.50	7.34	4.98	19.49	11.51	8.72	-	-	-	-	-	-
MonoGRNet [195] in [35]	11.90	7.56	5.76	19.72	12.81	10.15	47.59	32.28	25.50	52.13	35.99	28.72
MonoDIS [222] in [220]	11.06	7.60	6.37	18.45	12.58	10.66	-	-	-	-	-	-
M3D-RPN [15] in [17]	14.53	11.07	8.65	20.85	15.62	11.88	48.56	35.94	28.59	53.35	39.60	31.77
MoVi-3D [223]	14.28	11.13	9.68	22.36	17.87	15.73	-	-	-	-	-	-
MonoPair [35]	16.28	12.30	10.42	24.12	18.17	**15.76**	55.38	**42.39**	**37.99**	61.06	**47.63**	**41.92**
Kinematic (Image) [17]	18.28	13.55	10.13	25.72	18.82	14.48	54.70	39.33	31.25	60.87	44.36	34.48
Kinematic (Video) [17]	**19.76**	**14.10**	**10.47**	**27.83**	**19.72**	15.10	**55.44**	39.47	31.26	**61.79**	44.68	34.56
GrooMeD-NMS (Ours)	**19.67**	**14.32**	**11.27**	**27.38**	**19.75**	**15.92**	**55.62**	**41.07**	**32.89**	**61.83**	**44.98**	**36.29**



(a) Linear Scale

(b) Log Scale

Figure 2.3 **AP_{3D} Comparison at different depths and IoU_{3D} matching thresholds** on KITTI Val 1 Split.

object to a category based on its occlusion, truncation, and height in the image space. The AP $_{3D|R_{40}}$ performance on the Moderate category compares different models in the benchmark [67]. We focus primarily on the Car class following [17].

2.5.1 KITTI Test Mono3D

Tab. 2.2 summarizes the results of 3D object detection and BEV evaluation on KITTI Test Split. The results in Tab. 2.2 show that GrooMeD-NMS outperforms the baseline M3D-RPN [15] by a significant margin and several other SoTA methods on both the tasks. GrooMeD-NMS also outperforms augmentation based approach MoVi-3D [223] and depth-convolution based D4LCN [52]. Despite being an image-based method, GrooMeD-NMS performs competitively to the video-based method Kinematic (Video) [17], outperforming it on the most-challenging Hard set.

Table 2.4 **Comparisons with other NMS** on KITTI Val 1 cars ($IoU_{3D} \geq 0.7$). [Key: C= Classical, S= Soft-NMS [12], D= Distance-NMS [216], G= GrooMeD-NMS]

Method	Infer NMS	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
		Easy	Mod	Hard	Easy	Mod	Hard
Kinematic (Image)	C	18.28	13.55	10.13	25.72	18.82	14.48
Kinematic (Image)	S	18.29	13.55	10.13	25.71	18.81	14.48
Kinematic (Image)	D	18.25	13.53	10.11	25.71	18.82	14.48
Kinematic (Image)	G	18.26	13.51	10.10	25.67	18.77	14.44
GrooMeD-NMS	C	19.67	14.31	11.27	27.38	19.75	15.93
GrooMeD-NMS	S	19.67	14.31	11.27	27.38	19.75	15.93
GrooMeD-NMS	D	19.67	14.31	11.27	27.38	19.75	15.93
GrooMeD-NMS	G	19.67	14.32	11.27	27.38	19.75	15.92

2.5.2 KITTI Val 1 Mono3D

Results. Tab. 2.3 summarizes the results of 3D object detection and BEV evaluation on KITTI Val 1 Split at two IoU_{3D} thresholds of 0.7 and 0.5 [17, 35]. Tab. 2.3 results show that GrooMeD-NMS outperforms the baseline of M3D-RPN [15] and Kinematic (Image) [17] by a significant margin. Interestingly, GrooMeD-NMS (an image-based method) also outperforms the video-based method Kinematic (Video) [17] on most of the metrics. Thus, GrooMeD-NMS performs best on 6 out of the 12 cases (3 categories $\times$ 2 tasks $\times$ 2 thresholds) while second-best on all other cases. The performance is especially impressive since the biggest improvements are shown on the Moderate and Hard set, where objects are more distant and occluded.

AP_{3D} at different depths and IoU_{3D} thresholds. We next compare the AP_{3D} performance of GrooMeD-NMS and Kinematic (Image) on linear and log scale for objects at different depths of $[15, 30, 45, 60]$ meters and IoU_{3D} matching criteria of $0.3 \rightarrow 0.7$ in Fig. 2.3 as in [17]. Fig. 2.3 shows that GrooMeD-NMS outperforms the Kinematic (Image) [17] at all depths and all IoU_{3D} thresholds.

Comparisons with other NMS. We compare with the classical NMS, Soft-NMS [12] and Distance-NMS [216] in Tab. 2.4. More detailed results are in Tab. B.2 of the supplementary material. The results show that NMS inclusion in the training pipeline benefits the performance, unlike [12], which suggests otherwise. Training with GrooMeD-NMS helps because the network gets an additional signal through the GrooMeD-NMS layer whenever the best-localized box corresponding to an object is not selected. Interestingly, Tab. 2.4 also suggests that replacing GrooMeD-NMS

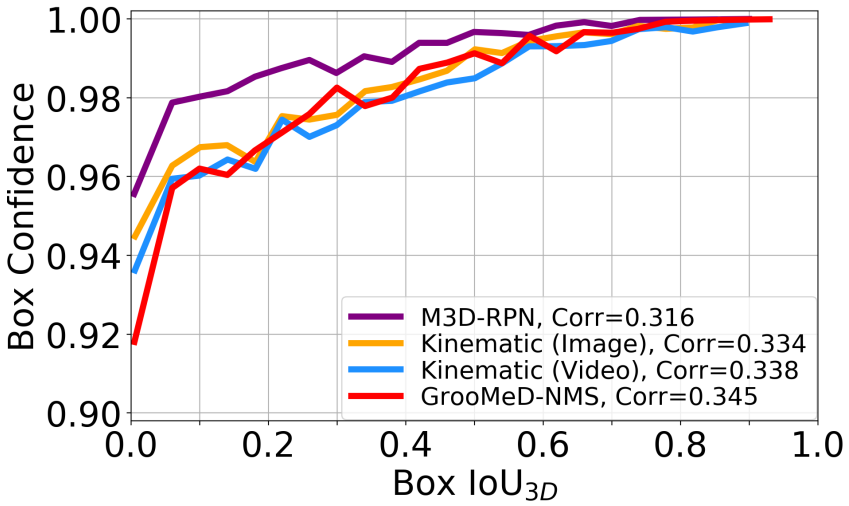


Figure 2.4 **Score-IoU$_{3D}$ plot** after the NMS. GrooMeD-NMS achieves the best correlation.

Table 2.5 **KITTI Val 2 cars** AP $_{3D|R_{40}}$ and AP $_{BEV|R_{40}}$ comparisons. [Key: **Best**, *= Released, †= Retrained].

Method	IoU$_{3D}$ ≥ 0.7						IoU$_{3D}$ ≥ 0.5					
	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)			AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
M3D-RPN [15]*	14.57	10.07	7.51	21.36	15.22	11.28	49.14	34.43	26.39	53.44	37.79	29.36
Kinematic (Image) [17]†	13.54	10.21	7.24	20.60	15.14	11.30	51.53	36.55	28.26	56.20	40.02	31.25
GrooMeD-NMS (Ours)	**14.72**	**10.87**	**7.67**	**22.03**	**16.05**	**11.93**	**51.91**	**36.78**	**28.40**	**56.29**	**40.31**	**31.39**

with the classical NMS in inference does not affect the performance.

Score-IoU$_{3D}$ Plot. We further correlate the scores with IoU$_{3D}$ after NMS of our model with two baselines - M3D-RPN [15] and Kinematic (Image) [17] and also the Kinematic (Video) [17] in Fig. 2.4. We obtain the best correlation of 0.345 exceeding the correlations of M3D-RPN, Kinematic (Image) and, also Kinematic (Video). This proves that including NMS in the training pipeline is beneficial.

Training and Inference Times. We now compare the training and inference times of including GrooMeD-NMS in the pipeline. Warmup training phase takes about 13 hours to train on a single 12 GB GeForce GTX Titan-X GPU. Full training phase of Kinematic (Image) and GrooMeD-NMS takes about 8 and 8.5 hours respectively. The inference time per image using classical and GrooMeD-NMS is 0.12 and 0.15 ms respectively. Tab. 2.4 suggests that changing the NMS from GrooMeD to classical during inference does not alter the performance. Then, the inference time of our method is the same as 0.12 ms.

Table 2.6 **Ablation studies** of GrooMeD-NMS on KITTI Val 1 cars.

Change from GrooMeD-NMS model:		$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
Changed	From ⟶ To	$AP_{3D\|R_{40}}$ (↑)			$AP_{BEV\|R_{40}}$ (↑)			$AP_{3D\|R_{40}}$ (↑)			$AP_{BEV\|R_{40}}$ (↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Training	Conf+NMS⟶No Conf+No NMS	16.66	12.10	9.40	23.15	17.43	13.48	51.47	38.58	30.98	56.48	42.53	34.37
	Conf+NMS⟶Conf+No NMS	19.16	13.89	10.96	27.01	19.33	14.84	57.12	41.07	32.79	61.60	44.58	35.97
	Conf+NMS⟶No Conf+NMS	15.02	11.21	8.83	21.07	16.27	12.77	48.01	36.18	29.96	53.82	40.94	33.35
Initialization	No Warmup	15.33	11.68	8.78	21.32	16.59	12.93	49.15	37.42	30.11	54.32	41.44	33.48
Pruning Function	Linear ⟶Exponential, $\tau = 1$	12.81	9.26	7.10	17.07	12.17	9.25	29.58	20.42	15.88	32.06	22.16	17.20
	Linear ⟶Exponential, $\tau = 0.5$ [12]	18.63	13.85	10.98	27.52	20.14	15.76	56.64	41.01	32.79	61.43	44.73	36.02
	Linear ⟶Exponential, $\tau = 0.1$	18.34	13.79	10.88	27.26	19.71	15.90	56.98	41.16	32.96	62.77	45.23	36.56
	Linear ⟶Sigmoidal, $\tau = 0.1$	17.40	13.21	9.80	26.77	19.26	14.76	55.15	40.77	32.63	60.56	44.23	35.74
Group+Mask	Group+Mask⟶No Group	18.43	13.91	11.08	26.53	19.46	15.83	55.93	40.98	32.78	61.02	44.77	36.09
	Group+Mask⟶Group+No Mask	18.99	13.74	10.24	26.71	19.21	14.77	55.21	40.69	32.55	61.74	44.67	36.00
Loss	Imagewise AP ⟶Vanilla AP	18.23	13.73	10.28	26.42	19.31	14.76	54.47	40.35	32.20	60.90	44.08	35.47
	Imagewise AP ⟶BCE	16.34	12.74	9.73	22.40	17.46	13.70	52.46	39.40	31.68	58.22	43.60	35.27
Inference NMS Scores	Class*Pred⟶Class	18.26	13.36	10.49	25.39	18.64	15.12	52.44	38.99	31.3	57.37	42.89	34.68
	Class*Pred⟶Pred	17.51	12.84	9.55	24.55	17.85	13.63	52.78	37.48	29.37	58.30	41.26	32.66
—	**GrooMeD-NMS (best model)**	**19.67**	**14.32**	**11.27**	**27.38**	**19.75**	**15.92**	**55.62**	**41.07**	**32.89**	**61.83**	**44.98**	**36.29**

2.5.3 KITTI Val 2 Mono3D

Tab. 2.5 summarizes the results of 3D object detection and BEV evaluation on KITTI Val 2 Split at two IoU_{3D} thresholds of 0.7 and 0.5 [17, 35]. Again, we use M3D-RPN [15] and Kinematic (Image) [17] as our baselines. We evaluate the released model of M3D-RPN [15] using the KITTI metric. [17] does not report Val 2 results, so we retrain on Val 2 using their public code. The results in Tab. 2.5 show that GrooMeD-NMS performs best in all cases. This is again impressive because the improvements are shown on Moderate and Hard set, consistent with Tabs. 2.2 and 2.3.

2.5.4 Ablation Studies on KITTI Val 1

Tab. 2.6 compares the modifications of our approach on KITTI Val 1 cars. Unless stated otherwise, we stick with the experimental settings described in Sec. 2.5. Using a confidence head (Conf+No NMS) proves beneficial compared to the warmup model (No Conf+No NMS), which is consistent with the observations of [17, 216]. Further, GrooMeD-NMS on classification scores (denoted by No Conf + NMS) is detrimental as the classification scores are not suited for localization [17,90]. Training the warmup model and then finetuning also works better than training without warmup as in [17] since the warmup phase allows GrooMeD-NMS to carry meaningful grouping of the boxes.

As described in Sec. 2.4.1.5, in addition to Linear, we compare two other functions for pruning function p: Exponential and Sigmoidal. Both of them do not perform as well as the Linear p possibly

because they have vanishing gradients close to overlap of zero or one. Grouping and masking both help our model to reach a better minimum. As described in Sec. 2.4.3, Imagewise AP loss is better than the Vanilla AP loss since it treats boxes of two images differently. Imagewise AP also performs better than the binary cross-entropy (BCE) loss proposed in [78, 80, 81, 192]. Using the product of self-balancing confidence and classification scores instead of using them individually as the scores to the NMS in inference is better, consistent with [99, 216, 241]. Class confidence performs worse since it does not have the localization information while the self-balancing confidence (Pred) gives the localization without considering whether the box belongs to foreground or background.

2.6 Conclusions

In this chapter, we present and integrate GrooMeD-NMS– a novel Grouped Mathematically Differentiable NMS for monocular 3D object detection, such that the network is trained end-to-end with a loss on the boxes after NMS. We first formulate NMS as a matrix operation and then do unsupervised grouping and masking of the boxes to obtain a simple closed-form expression of the NMS. GrooMeD-NMS addresses the mismatch between training and inference pipelines and, therefore, forces the network to select the best 3D box in a differentiable manner. As a result, GrooMeD-NMS achieves state-of-the-art monocular 3D object detection results on the KITTI benchmark dataset. Although our implementation demonstrates monocular 3D object detection, GrooMeD-NMS is fairly generic for other object detection tasks. Future work includes applying this method to tasks such as LiDAR-based 3D object detection and pedestrian detection.

Limitation. GrooMeD-NMS does not fully solve the generalization issue.

CHAPTER 3

DEVIANT: DEPTH EQUIVARIANT NETWORK FOR MONOCULAR 3D OBJECT DETECTION

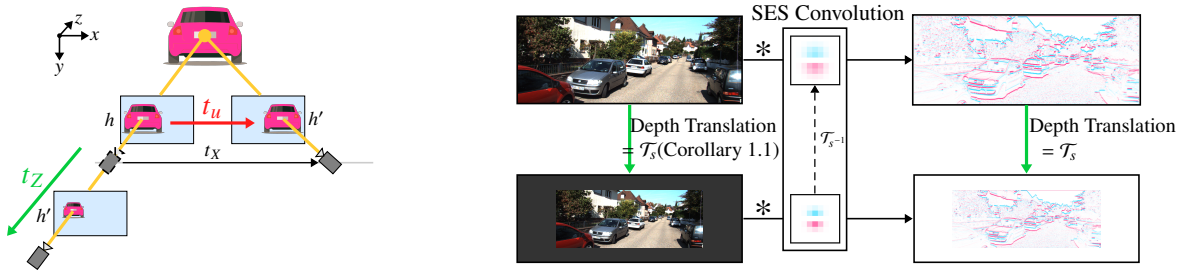
Modern neural networks use building blocks such as convolutions that are equivariant to arbitrary 2D translations in the Euclidean manifold. However, these vanilla blocks are not equivariant to arbitrary 3D translations in the projective manifold. Even then, all monocular 3D detectors use vanilla blocks to obtain the 3D coordinates, a task for which the vanilla blocks are not designed for. This chapter takes the first step towards convolutions equivariant to arbitrary 3D translations in the projective manifold. Since the depth is the hardest to estimate for monocular detection, this chapter proposes Depth Equivariant Network (DEVIANT) built with existing scale equivariant steerable blocks. As a result, DEVIANT is equivariant to the depth translations in the projective manifold whereas vanilla networks are not. The additional depth equivariance forces the DEVIANT to learn consistent depth estimates, and therefore, DEVIANT achieves state-of-the-art monocular 3D detection results on KITTI and Waymo datasets in the image-only category and performs competitively to methods using extra information. Moreover, DEVIANT works better than vanilla networks in cross-dataset evaluation.

3.1 Introduction

Monocular 3D object detection is a fundamental task in computer vision, where the task is to infer 3D information including depth from a single monocular image. It has applications in augmented reality [2], gaming [201], robotics [213], and more recently in autonomous driving [15, 220] as a fallback solution for LiDAR.

Most of the monocular 3D methods attach extra heads to the 2D Faster-RCNN [202] or CenterNet [306] for 3D detections. Some change architectures [123, 143, 232] or losses [15, 35]. Others incorporate augmentation [223], or confidence [17, 143]. Recent ones use in-network ensembles [159, 301] for better depth estimation.

Most of these methods use vanilla blocks such as convolutions that are *equivariant* to arbitrary 2D translations [19, 198]. In other words, whenever we shift the ego camera in 2D (See t_u of



(a) Idea.

(b) Depth Equivariance.

Figure 3.1 **(a) Idea.** Vanilla CNN is equivariant to projected 2D translations t_u, t_v (in red) of the ego camera. The ego camera moves in 3D in driving scenes which breaks this assumption. We propose DEVIANT which is additionally equivariant to depth translations t_Z (in green) in the projective manifold. **(b) Depth Equivariance**. DEVIANT enforces additional consistency among the feature maps of an image and its transformation caused by the ego depth translation. $\mathcal{T}_s$ = scale transformation, $*$ = vanilla convolution.

Fig. 3.1), the new image (projection) is a translation of the original image, and therefore, these methods output a translated feature map. However, in general, the camera moves in depth in driving scenes instead of 2D (See t_Z of Fig. 3.1). So, the new image is not a translation of the original input image due to the projective transform. Thus, using vanilla blocks in monocular methods is a mismatch between the assumptions and the regime where these blocks operate. Additionally, there is a huge generalization gap between training and validation for monocular 3D detection (See Modeling translation equivariance in the correct manifold improves generalization for tasks in spherical [41] and hyperbolic [64] manifolds. Monocular detection involves processing pixels (3D point projections) to obtain the 3D information, and is thus a task in the projective manifold. Moreover, the depth in monocular detection is ill-defined [232], and thus, the hardest to estimate [166]. Hence, using building blocks *equivariant to depth translations in the projective manifold* is a natural choice for improving generalization and is also at the core of this work (See Sec. C.1.8).

Recent monocular methods use flips [15], scale [159, 223], mosaic [11, 238] or copy-paste [135] augmentation, depth-aware convolution [15], or geometry [151, 159, 218, 302] to improve generalization. Although all these methods improve performance, a major issue is that their backbones are not designed for the projective world. This results in the depth estimation going haywire with a slight ego movement [307]. Moreover, data augmentation, *e.g.*, flips, scales,

Table 3.1 **Equivariance comparisons**. [Key: Proj.= Projected, ax= axis]

Translation →	3D			Proj. 2D	
	$x-$ax (t_X)	$y-$ax (t_Y)	$z-$ax (t_Z)	u-ax (t_u)	v-ax (t_v)
Vanilla CNN	–	–	–	✓	✓
Log-polar [313]	–	–	✓	–	–
DEVIANT	–	–	✓	✓	✓
Ideal	✓	✓	✓	–	–

mosaic, copy-paste, is not only limited for the projective tasks, but also does not guarantee desired behavior [63].

To address the mismatch between assumptions and the operating regime of the vanilla blocks and improve generalization, we take the first step towards convolutions equivariant to arbitrary 3D translations in the projective manifold. We propose Depth Equivariant Network (DEVIANT) which is additionally equivariant to depth translations in the projective manifold as shown in Tab. 3.1. Building upon the classic result from [76], we simplify it under reasonable assumptions about the camera movement in autonomous driving to get scale transformations. The scale equivariant blocks are well-known in the literature [68, 92, 227, 309], and consequently, we replace the vanilla blocks in the backbone with their scale equivariant steerable counterparts [227] to additionally embed equivariance to depth translations in the projective manifold. Hence, DEVIANT learns consistent depth estimates and improves monocular detection.

In summary, the main contributions of this work include:

- We study the modeling error in monocular 3D detection and propose depth equivariant networks built with scale equivariant steerable blocks as a solution.
- We achieve state-of-the-art (SoTA) monocular 3D object detection results on the KITTI and Waymo datasets in the image-only category and perform competitively to methods which use extra information.
- We experimentally show that DEVIANT works better in cross-dataset evaluation suggesting better generalization than vanilla CNN backbones.

Table 3.2 Equivariances known in the literature.

Transformation → / **Manifold ↓**	Translation	Rotation	Scale	Flips	Learned
Euclidean	Vanilla CNN [115]	Polar, Steerable [266]	Log-polar [79], Steerable [68]	ChiralNets [288]	Transformers [55]
Spherical	Spherical CNN [41]	–	–	–	–
Hyperbolic	Hyperbolic CNN [64]	–	–	–	–
Projective	Monocular Detector	–	–	–	–

3.2 Related Works

Equivariant Neural Networks. The success of convolutions in CNN has led people to look for their generalizations [43, 262]. Convolution is the unique solution to 2D translation equivariance in the Euclidean manifold [19, 20, 198]. Thus, convolution in CNN is a prior in the Euclidean manifold. Several works explore other group actions in the Euclidean manifold such as 2D rotations [42, 50, 171, 263], scale [98, 170], flips [288], or their combinations [247, 266]. Some consider 3D translations [265] and rotations [239]. Few [55, 264, 304] attempt learning the equivariance from the data, but such methods have significantly higher data requirements [265]. Others change the manifold to spherical [41], hyperbolic [64], graphs [173], or arbitrary manifolds [97]. Monocular 3D detection involves operations on pixels which are projections of 3D point and thus, works in a different manifold namely projective manifold. Tab. 3.2 summarizes all these equivariances known thus far.

Scale Equivariant Networks. Scale equivariance in the Euclidean manifold is more challenging than the rotations because of its acyclic and unbounded nature [198]. There are two major lines of work for scale equivariant networks. The first [56, 79] infers the global scale using log-polar transform [313], while the other infers the scale locally by convolving with multiple scales of images [98] or filters [278]. Several works [68, 92, 227, 309] extend the local idea, using steerable filters [62]. Another work [267] constructs filters for integer scaling. We compare the two kinds of scale equivariant convolutions on the monocular 3D detection task and show that steerable convolutions are better suited to embed depth (scale) equivariance. Scale equivariant networks have been used for classification [56, 68, 227], 2D tracking [226] and 3D object classification [56]. We are the first to use scale equivariant networks for monocular 3D detection.

3D Object Detection. Accurate 3D object detection uses sparse data from LiDARs [215], which are expensive and do not work well in severe weather [232] and glassy environments. Hence, several works have been on monocular camera-based 3D object detection, which is simplistic but has scale/depth ambiguity [232]. Earlier approaches [31, 61, 186, 187] use hand-crafted features, while the recent ones use deep learning. Some change architectures [123, 143, 146, 232] or losses [15, 35]. Some use scale [159, 223], mosaic [238] or copy-paste [135] augmentation. Others incorporate depth in convolution [15, 52], or confidence [17, 111, 143]. More recent ones use in-network ensembles to predict the depth deterministically [301] or probabilistically [159]. A few use temporal cues [17], NMS [109], or corrected camera extrinsics [307] in the training pipeline. Some also use CAD models [25, 154] or LiDAR [199] in training. Another line of work called Pseudo-LiDAR [162, 165, 181, 221, 254] estimates the depth first, and then uses a point cloud-based 3D object detector. We refer to [163] for a detailed survey. Our work is the first to use scale equivariant blocks in the backbone for monocular 3D detection.

3.3 Background

We first provide the necessary definitions which are used throughout this chapter. These are not our contributions and can be found in the literature [21, 76, 265].

Equivariance. Consider a group of transformations G, whose individual members are g. Assume Φ denote the mapping of the inputs h to the outputs y. Let the inputs and outputs undergo the transformation $\mathcal{T}_g^h$ and $\mathcal{T}_g^y$ respectively. Then, the mapping Φ is equivariant to the group G [265] if $\Phi(\mathcal{T}_g^h h) = \mathcal{T}_g^y(\Phi h), \forall\ g \in G$. Thus, equivariance provides an explicit relationship between input transformations and feature-space transformations at each layer of the neural network [265], and intuitively makes the learning easier. The mapping Φ is the vanilla convolution when the $\mathcal{T}_g^h = \mathcal{T}_g^y = \mathcal{T}_\mathbf{t}$ where $\mathcal{T}_\mathbf{t}$ denotes the translation $\mathbf{t}$ on the discrete grid [19, 20, 198]. These vanilla convolution introduce weight-tying [115] in fully connected neural networks resulting in a greater generalization. A special case of equivariance is the invariance [265] which is given by $\Phi(\mathcal{T}_g^h h) = \Phi h, \forall\ g \in G$.

Projective Transformations. Our idea is to use equivariance to depth translations in the projective

manifold since the monocular detection task belongs to this manifold. A natural question to ask is whether such equivariants exist in the projective manifold. [21] answers this question in negative, and says that such equivariants do not exist in general. However, such equivariants exist for special classes, such as planes. An intuitive way to understand this is to infer the rotations and translations by looking at the two projections (images). For example, the result of [21] makes sense if we consider a car with very different front and back sides as in Fig. C.2. A 180° ego rotation around the car means the projections (images) are its front and the back sides, which are different. Thus, we can not infer the translations and rotations from these two projections. Based on this result, we stick with **locally** planar objects *i.e.* we assume that a 3D object is made of several *patch planes*. (See last row of Fig. 3.2b as an example). It is important to stress that we do **NOT** assume that the 3D object such as car is planar. The local planarity also agrees with the property of manifolds that manifolds locally resemble n-dimensional Euclidean space and because the projective transform maps planes to planes, the patch planes in 3D are also locally planar. We show a sample planar patch and the 3D object in Fig. C.1 in the appendix.

Planarity and Projective Transformation. Example 13.2 from [76] links the planarity and projective transformations. Although their result is for stereo with two different cameras ($\mathbf{K}, \mathbf{K}'$), we substitute $\mathbf{K}=\mathbf{K}'$ to get Th. 1.

Theorem 1. *[76] Consider a 3D point lying on a patch plane $mx+ny+oz+p=0$, and observed by an ego camera in a pinhole setup to give an image h. Let $\mathbf{t}=(t_X, t_Y, t_Z)$ and $\mathbf{R}=[r_{ij}]_{3\times3}$ denote a translation and rotation of the ego camera respectively. Observing the same 3D point from a new camera position leads to an image h'. Then, the image h is related to the image h' by the projective transformation*

$$\mathcal{T} : h(u-u_0, v-v_0) = \tag{3.1}$$
$$h'\left(f\frac{\left(r_{11}+\bar{t}_X\frac{m}{p}\right)(u-u_0)+\left(r_{21}+\bar{t}_X\frac{n}{p}\right)(v-v_0)+\left(r_{31}+\bar{t}_X\frac{o}{p}\right)f}{\left(r_{13}+\bar{t}_Z\frac{m}{p}\right)(u-u_0)+\left(r_{23}+\bar{t}_Z\frac{n}{p}\right)(v-v_0)+\left(r_{33}+\bar{t}_Z\frac{o}{p}\right)f},\right.$$

$$f\frac{\left(r_{12}+\bar{t}_Y\frac{m}{p}\right)(u-u_0)+\left(r_{22}+\bar{t}_Y\frac{n}{p}\right)(v-v_0)+\left(r_{32}+\bar{t}_Y\frac{o}{p}\right)f}{\left(r_{13}+\bar{t}_Z\frac{m}{p}\right)(u-u_0)+\left(r_{23}+\bar{t}_Z\frac{n}{p}\right)(v-v_0)+\left(r_{33}+\bar{t}_Z\frac{o}{p}\right)f},$$

where f and (u_0, v_0) denote the focal length and principal point of the ego camera, and $(\bar{t}_X, \bar{t}_Y, \bar{t}_Z) = \mathbf{R}^T\mathbf{t}$.

3.4 Depth Equivariant Backbone

The projective transformation in Eq. (3.1) from [76] is complicated and also involves rotations, and we do not know which convolution obeys this projective transformation. Hence, we simplify Eq. (3.1) under reasonable assumptions to obtain a familiar transformation for which the *convolution* is known.

Corollary 1.1. *When the ego camera translates in depth without rotations* ($\mathbf{R} = \mathbf{I}$)*, and the patch plane is "approximately" parallel to the image plane, the image h locally is a scaled version of the second image h' independent of focal length, i.e.*

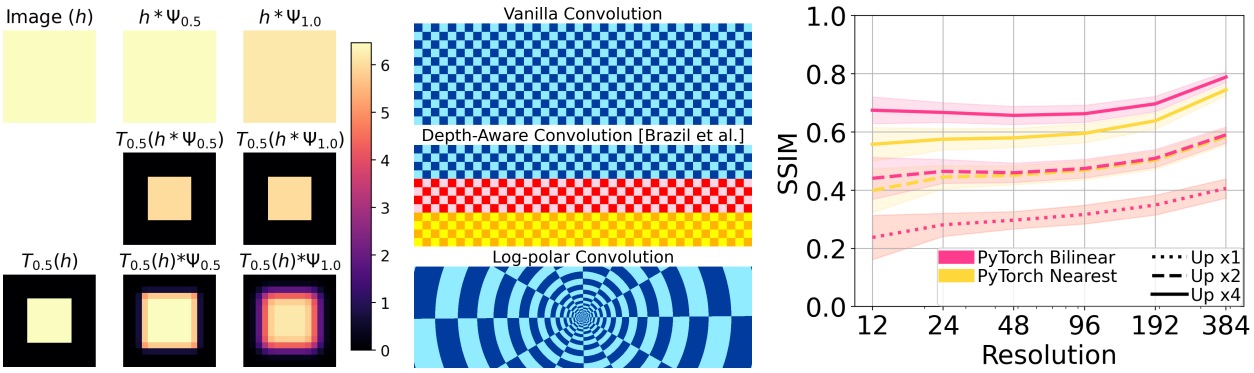
$$\mathcal{T}_s : h(u-u_0, v-v_0) \approx h'\left(\frac{u-u_0}{1+t_Z\frac{o}{p}}, \frac{v-v_0}{1+t_Z\frac{o}{p}}\right). \tag{3.2}$$

where f and (u_0, v_0) denote the focal length and principal point of the ego camera, and t_Z denotes the ego translation.

See Sec. C.1.6 for the detailed explanation of Corollary 1.1. Corollary 1.1 says

$$\mathcal{T}_s : h(u-u_0, v-v_0) \approx h'\left(\frac{u-u_0}{s}, \frac{v-v_0}{s}\right), \tag{3.3}$$

where, $s = 1 + t_Z\frac{o}{p}$ denotes the scale and $\mathcal{T}_s$ denotes the scale transformation. The scale $s < 1$ suggests downscaling, while $s > 1$ suggests upscaling. Corollary 1.1 shows that the transformation $\mathcal{T}_s$ is independent of the focal length and that scale is a linear function of the depth translation. Hence, the depth translation in the projective manifold induces scale transformation and thus, the depth equivariance in the projective manifold is the scale equivariance in the Euclidean manifold. Mathematically, the desired equivariance is $[\mathcal{T}_s(h) * \Psi] = \mathcal{T}_s[h * \Psi_{s^{-1}}]$, where Ψ denotes the filter (See Sec. C.1.7). As CNN is not a scale equivariant (SE) architecture [227], we aim to get SE backbone which makes the architecture equivariant to depth translations in the projective



(a) SES Convolution Output. (b) Receptive fields. (c) Log-polar SSIM.

Figure 3.2 **(a) Scale Equivariance**. We apply SES convolution [227] with two scales on a single channel toy image h. **(b) Receptive fields** of convolutions in the Euclidean manifold. Colors represent different weights, while shades represent the same weight. **(c) Impact of discretization on log-polar convolution.** SSIM is very low at small resolutions and is not 1 even after upscaling by 4. [Key: Up= Upscaling]

manifold. The scale transformation is a familiar transformation and SE convolutions are well known [68, 92, 227, 309].

Scale Equivariant Steerable (SES) Blocks. We use the existing SES blocks [226, 227] to construct our Depth Equivariant Network (DEVIANT) backbone. As [226] does not construct SE-DLA-34 backbones, we construct our DEVIANT backbone as follows. We replace the vanilla convolutions by the SES convolutions [226] with the basis as Hermite polynomials. SES convolutions result in multi-scale representation of an input tensor. As a result, their output is five-dimensional instead of four-dimensional. Thus, we replace the 2D pools and batch norm (BN) by 3D pools and 3D BN respectively. The Scale-Projection layer [227] carries a max over the extra (scale) dimension to project five-dimensional tensors to four dimensions (See Fig. C.5) in the supplementary). Ablation in Sec. 3.5.3 confirms that BN and Pool (BNP) should also be SE for the best performance.

The SES convolutions [68, 227, 309] are based on steerable-filters [62]. Steerable approaches [68] first pre-calculate the non-trainable multi-scale basis in the Euclidean manifold and then build filters by the linear combinations of the trainable weights $\mathbf{w}$. The number of trainable weights $\mathbf{w}$ equals the number of filters at one particular scale. The linear combination of multi-scale basis ensures that the filters are also multi-scale. Thus, SES blocks bypass grid conversion and do not suffer from sampling effects.

We show the convolution of toy image h with a SES convolution in Fig. 3.2a. Let Ψ_s denote

the filter at scale s. The convolution between downscaled image and filter $\mathcal{T}_{0.5}(h) * \Psi_{0.5}$ matches the downscaled version of original image convolved with upscaled filter $\mathcal{T}_{0.5}(h * \Psi_{1.0})$. Fig. 3.2a (right column) shows that the output of a CNN exhibits aliasing in general and is therefore, not scale equivariant.

Log-polar Convolution: Impact of Discretization. An alternate way to convert the depth translation t_Z of Eq. (3.2) to shift is by converting the images to log-polar space [313] around the principal point (u_0, v_0), as

$$h(\ln r, \theta) \approx h' \left(\ln r - \ln \left(1 + t_Z \frac{o}{p} \right), \ \theta \right), \tag{3.4}$$

with $r = \sqrt{(u-u_0)^2 + (v-v_0)^2}$, and $\theta = \tan^{-1}\left(\frac{v-v_0}{u-u_0}\right)$. The log-polar transformation converts the scale to translation, so using convolution in the log-polar space is equivariant to the logarithm of the depth translation t_Z. We show the receptive field of log-polar convolution in Fig. 3.2b. The log-polar convolution uses a smaller receptive field for objects closer to the principal point, while a larger field away from the principal point. We implemented log-polar convolution and found that its performance (See Tab. 3.11) is not acceptable, consistent with [227]. We attribute this behavior to the discretization of pixels and loss of 2D translation equivariance. Eq. (3.4) is perfectly valid in the continuous world (Note the use of parentheses instead of square brackets in Eq. (3.4)). However, pixels reside on discrete grids, which gives rise to sampling errors [112]. We discuss the impact of discretization on log-polar convolution in Sec. 3.5.2 and show it in Fig. 3.2c. Hence, we do not use log-polar convolution for the DEVIANT backbone.

Comparison of Equivariance s for Monocular 3D Detection. We now compare equivariances for monocular 3D detection task. An ideal monocular detector should be equivariant to arbitrary 3D translations (t_X, t_Y, t_Z). However, most monocular detectors [109, 159] estimate 2D projections of 3D centers and the depth, which they back-project in 3D world via known camera intrinsics. Thus, a good enough detector shall be equivariant to 2D translations (t_u, t_v) for projected centers as well as equivariant to depth translations (t_Z).

Existing detector backbones [109, 159] are only equivariant to 2D translations as they use vanilla convolutions that produce 4D feature maps. Log-polar backbones is equivariant to logarithm of

depth translations but not to 2D translations. DEVIANT uses SES convolutions to produce 5D feature maps. The extra dimension in 5D feature map captures the changes in scale (for depth), while these feature maps individually are equivariant to 2D translations (for projected centers). Hence, DEVIANT augments the 2D translation equivariance (t_u, t_v) of the projected centers with the depth translation equivariance. We emphasize that although DEVIANT is **not** equivariant to arbitrary 3D translations in the projective manifold, DEVIANT **does** provide the equivariance to depth translations (t_Z) and is thus a first step towards the ideal equivariance. Our experiments (Sec. 3.5) show that even this additional equivariance benefits monocular 3D detection task. This is expected because depth is the hardest parameter to estimate [166]. Tab. 3.1 summarizes these equivariances. Moreover, Tab. 3.10 empirically shows that 2D detection does not suffer and therefore, confirms that DEVIANT indeed augments the 2D equivariance with the depth equivariance. An idea similar to DEVIANT is the optical expansion [280] which augments optical flow with the scale information and benefits depth estimation.

3.5 Experiments

Our experiments use the KITTI [67], Waymo [230] and nuScenes datasets [22]. We modify the publicly-available PyTorch [184] code of GUP Net [159] and use the GUP Net model as our baseline. For DEVIANT, we keep the number of scales as three [226]. DEVIANT takes 8.5 hours for training and 0.04s per image for inference on a single A100 GPU.

Evaluation Metrics. KITTI evaluates on three object categories: Easy, Moderate and Hard. It assigns each object to a category based on its occlusion, truncation, and height in the image space. KITTI uses AP $_{3D|R_{40}}$ percentage metric on the Moderate category to benchmark models [67] following [220, 222].

Waymo evaluates on two object levels: Level_1 and Level_2. It assigns each object to a level based on the number of LiDAR points included in its 3D box. Waymo uses APH_{3D} percentage metric which is the incorporation of heading information in AP_{3D} to benchmark models. It also provides evaluation at three distances $[0, 30)$, $[30, 50)$ and $[50, \infty)$ meters.

Data Splits. We use the following splits of the KITTI,Waymo and nuScenes:

Table 3.3 **Results on KITTI Test cars** at $\text{IoU}_{3D} \geq 0.7$. Previous results are from the leader-board or papers. We show 3 methods in each Extra category and 6 methods in the image-only category. [Key: **Best**, **Second Best**]

Method	Extra	AP $_{3D\|R_{40}}$ [%](↑)			AP $_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard
AutoShape [154]	CAD	22.47	14.17	11.36	30.66	20.08	15.59
PCT [246]	Depth	21.00	13.37	11.31	29.65	19.03	15.92
DFR-Net [312]	Depth	19.40	13.63	10.35	28.17	19.17	14.84
MonoDistill [39]	Depth	22.97	16.03	13.60	31.87	22.59	19.72
PatchNet-C [221]	LiDAR	22.40	12.53	10.60	–	–	–
CaDDN [199]	LiDAR	19.17	13.41	11.46	27.94	18.91	17.19
DD3D [181]	LiDAR	23.22	16.34	14.20	30.98	22.56	20.03
MonoEF [307]	Odometry	21.29	13.87	11.71	29.03	19.70	17.26
Kinematic [17]	Video	19.07	12.72	9.17	26.69	17.52	13.10
GrooMeD-NMS [109]	–	18.10	12.32	9.65	26.19	18.27	14.05
MonoRCNN [218]	–	18.36	12.65	10.03	25.48	18.11	14.10
MonoDIS-M [220]	–	16.54	12.97	11.04	24.45	19.25	16.87
Ground-Aware [151]	–	**21.65**	13.25	9.91	**29.81**	17.98	13.08
MonoFlex [301]	–	19.94	13.89	**12.07**	28.23	**19.75**	**16.89**
GUP Net [159]	–	20.11	**14.20**	11.77	–	–	–
DEVIANT (Ours)	–	**21.88**	**14.46**	**11.89**	**29.65**	**20.44**	**17.43**

- *KITTI Test (Full) split*: Official KITTI 3D benchmark [1] consists of 7,481 training and 7,518 testing images [67].
- *KITTI Val split*: It partitions the 7,481 training images into 3,712 training and 3,769 validation images [32].
- *Waymo Val split*: This split [199, 246] contains 52,386 training and 39,848 validation images from the front camera. We construct its training set by sampling every third frame from the training sequences as in [199, 246].
- *nuScenes Val split:* It consists of 28,130 training and 6,019 validation images from the front camera [22]. We use this split for evaluation [218].

3.5.1 KITTI Test Mono3D

Cars. Tab. 3.3 lists out the results of monocular 3D detection and BEV evaluation on KITTI Test cars. Tab. 3.3 results show that DEVIANT outperforms the GUP Net and several other SoTA methods on both tasks. Except DD3D [181] and MonoDistill [39], DEVIANT, an image-based method, also outperforms other methods that use extra information.

Cyclists and Pedestrians. Tab. 3.4 lists out the results of monocular 3D detection on KITTI Test

Table 3.4 **Results on KITTI Test cyclists and pedestrians** (Cyc/Ped) at $\text{IoU}_{3D} \geq 0.5$. Previous results are from the leader-board or papers. [Key: **Best**, **Second Best**]

Method	Extra	Cyc AP $_{3D\|R_{40}}$ [%](↑)			Ped AP $_{3D\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard
DDMP-3D [245]	Depth	4.18	2.50	2.32	4.93	3.55	3.01
DFR-Net [312]	Depth	5.69	3.58	3.10	6.09	3.62	3.39
MonoDistill [39]	Depth	5.53	2.81	2.40	12.79	8.17	7.45
CaDDN [199]	LiDAR	7.00	3.41	3.30	12.87	8.14	6.76
DD3D [181]	LiDAR	2.39	1.52	1.31	13.91	9.30	8.05
MonoEF [307]	Odometry	1.80	0.92	0.71	4.27	2.79	2.21
MonoDIS-M [220]	–	1.17	0.54	0.48	7.79	5.14	4.42
MonoFlex [301]	–	3.39	2.10	1.67	11.89	8.16	6.81
GUP Net [159]	–	**4.18**	**2.65**	**2.09**	**14.72**	**9.53**	**7.87**
DEVIANT (Ours)	–	**5.05**	**3.13**	**2.59**	**13.43**	**8.65**	**7.69**

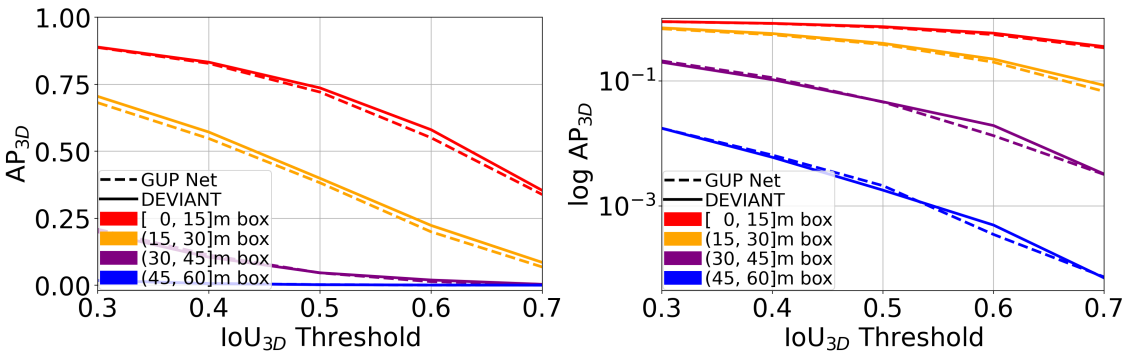
Table 3.5 **Results on KITTI Val cars**. Comparison with bigger CNN backbones in Tab. C.4. [Key: **Best**, **Second Best**, $^{-}$= No pretrain]

Method	Extra	$\text{IoU}_{3D} \geq 0.7$						$\text{IoU}_{3D} \geq 0.5$					
		AP $_{3D\|R_{40}}$ [%](↑)			AP $_{BEV\|R_{40}}$ [%](↑)			AP $_{3D\|R_{40}}$ [%](↑)			AP $_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
DDMP-3D [245]	Depth	28.12	20.39	16.34	–	–	–	–	–	–	–	–	–
PCT [246]	Depth	38.39	27.53	24.44	47.16	34.65	28.47	–	–	–	–	–	–
MonoDistill [39]	Depth	24.31	18.47	15.76	33.09	25.40	22.16	65.69	49.35	43.49	71.45	53.11	46.94
CaDDN [199]	LiDAR	23.57	16.31	13.84	–	–	–	–	–	–	–	–	–
PatchNet-C [221]	LiDAR	24.51	17.03	13.25	–	–	–	–	–	–	–	–	–
DD3D (DLA34) [181]	LiDAR	–	–	–	33.5	26.0	22.6	–	–	–	–	–	–
DD3D $^{-}$(DLA34) [181]	LiDAR	–	–	–	26.8	20.2	16.7	–	–	–	–	–	–
MonoEF [307]	Odometry	18.26	16.30	15.24	26.07	25.21	21.61	57.98	51.80	49.34	63.40	61.13	53.22
Kinematic [17]	Video	19.76	14.10	10.47	27.83	19.72	15.10	55.44	39.47	31.26	61.79	44.68	34.56
MonoRCNN [218]	–	16.61	13.19	10.65	25.29	19.22	15.30	–	–	–	–	–	–
MonoDLE [166]	–	17.45	13.66	11.68	24.97	19.33	17.01	55.41	43.42	37.81	60.73	46.87	41.89
GrooMeD-NMS [109]	–	19.67	14.32	11.27	27.38	19.75	15.92	55.62	41.07	32.89	61.83	44.98	36.29
Ground-Aware [151]	–	23.63	16.16	12.06	–	–	–	**60.92**	42.18	32.02	–	–	–
MonoFlex [301]	–	**23.64**	**17.51**	**14.83**	–	–	–	–	–	–	–	–	–
GUP Net (Reported) [159]	–	22.76	16.46	13.72	**31.07**	**22.94**	**19.75**	57.62	42.33	37.59	61.78	47.06	40.88
GUP Net (Retrained) [159]	–	21.10	15.48	12.88	28.58	20.92	17.83	58.95	**43.99**	**38.07**	**64.60**	**47.76**	**42.97**
DEVIANT (Ours)	–	**24.63**	**16.54**	**14.52**	**32.60**	**23.04**	**19.99**	**61.00**	**46.00**	**40.18**	**65.28**	**49.63**	**43.50**

Cyclist and Pedestrians. The results show that DEVIANT achieves SoTA results in the image-only category on the challenging Cyclists, and is competitive on Pedestrians.

3.5.2 KITTI Val Mono3D

Cars. Tab. 3.5 summarizes the results of monocular 3D detection and BEV evaluation on KITTI Val split at two IoU_{3D} thresholds of 0.7 and 0.5 [35, 109]. We report the **median** model over 5 runs. The results show that DEVIANT outperforms the GUP Net [159] baseline by a significant margin. The biggest improvements shows up on the Easy set. Significant improvements are also



(a) Linear Scale (b) Log Scale

Figure 3.3 **AP_{3D} at different depths and IoU_{3D} thresholds** on KITTI Val Split.

Table 3.6 **Cross-dataset evaluation** of the KITTI Val model on KITTI Val and nuScenes frontal Val cars with depth MAE (↓). [Key: **Best**, **Second Best**]

Method	KITTI Val 1				nuScenes frontal Val			
	0−20	20−40	40−∞	All	0−20	20−40	40−∞	All
M3D-RPN [15]	0.56	1.33	2.73	1.26	0.94	3.06	10.36	2.67
MonoRCNN [218]	0.46	1.27	2.59	1.14	0.94	2.84	8.65	2.39
GUP Net [159]	**0.45**	**1.10**	**1.85**	**0.89**	**0.82**	**1.70**	**6.20**	**1.45**
DEVIANT	**0.40**	**1.09**	**1.80**	**0.87**	**0.76**	**1.60**	**4.50**	**1.26**

on the Moderate and Hard sets. Interestingly, DEVIANT also outperforms DD3D [181] by a large margin when the large-dataset pretraining is not done (denoted by DD3D $^{-}$).

AP_{3D} at different depths and IoU_{3D} thresholds. We next compare the AP_{3D} of DEVIANT and GUP Net in Fig. 3.3 at different distances in meters and IoU_{3D} matching criteria of $0.3 \rightarrow 0.7$ as in [109]. Fig. 3.3 shows that DEVIANT is effective over GUP Net [159] at all depths and higher IoU_{3D} thresholds.

Cross-Dataset Evaluation. Tab. 3.6 shows the result of our KITTI Val model on the KITTI Val and nuScenes [22] frontal Val images, using mean absolute error (MAE) of the depth of the boxes [218]. More details are in Sec. C.3.1. DEVIANT outperforms GUP Net on most of the metrics on both the datasets, which confirms that DEVIANT generalizes better than CNNs. DEVIANT performs exceedingly well in the cross-dataset evaluation than [15, 159, 218]. We believe this happens because [15, 159, 218] rely on data or geometry to get the depth, while DEVIANT is equivariant to the depth translations, and therefore, outputs consistent depth. So, DEVIANT is more robust to data distribution changes.

Table 3.7 **Scale Augmentation vs Scale Equivariance** on KITTI Val cars. [Key: **Best**, Eqv= Equivariance, Aug= Augmentation]

Method	Scale Eqv	Scale Aug	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)		
			Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [159]			20.82	14.15	12.44	29.93	20.90	17.87	**62.37**	44.40	39.61	**66.81**	48.09	43.14
		✓	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
DEVIANT	✓		21.33	14.77	12.57	28.79	20.28	17.59	59.31	43.25	37.64	63.94	47.02	41.12
	✓	✓	**24.63**	**16.54**	**14.52**	**32.60**	**23.04**	**19.99**	61.00	**46.00**	**40.18**	65.28	**49.63**	**43.50**

Table 3.8 **Comparison of Equivariant Architectures** on KITTI Val cars. [Key: **Best**, Eqv= Equivariance, †= Retrained]

Method	Eqv	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
		$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
DETR3D† [257]	Learned	1.94	1.26	1.09	4.41	3.06	2.79	20.09	13.80	12.78	26.51	18.49	17.36
GUP Net [159]	2D	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
DEVIANT	2D +Depth	**24.63**	**16.54**	**14.52**	**32.60**	**23.04**	**19.99**	**61.00**	**46.00**	**40.18**	**65.28**	**49.63**	**43.50**

Alternatives to Equivariance. We now compare with alternatives to equivariance in the following paragraphs.

(a) Scale Augmentation. A withstanding question in machine learning is the choice between equivariance and data augmentation [63]. Tab. 3.7 compares scale equivariance and scale augmentation. GUP Net [159] uses scale-augmentation and therefore, Tab. 3.7 shows that equivariance also benefits models which use scale-augmentation. This agrees with Tab. 2 of [227], where they observe that both augmentation and equivariance benefits classification on MNIST-scale dataset.

(b) Other Equivariant Architectures. We now benchmark adding depth (scale) equivariance to a 2D translation equivariant CNN and a transformer which learns the equivariance. Therefore, we compare DEVIANT with GUP Net [159] (a CNN), and DETR3D [257] (a transformer) in Tab. 3.8. As DETR3D does not report KITTI results, we trained DETR3D on KITTI using their public code. DEVIANT outperforms GUP Net and also surpasses DETR3D by a large margin. This happens because learning equivariance requires more data [265] compared to architectures which hardcode equivariance like CNN or DEVIANT.

(c) Dilated Convolution. DEVIANT adjusts the receptive field based on the object scale, and so, we compare with the dilated CNN (DCNN) [291] and D4LCN [52] in Tab. 3.9. The results

Table 3.9 **Comparison with Dilated Convolution** on KITTI Val cars. [Key: **Best**]

Method	Extra	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
		$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
D4LCN [52]	Depth	22.32	16.20	12.30	31.53	22.58	17.87	–	–	–	–	–	–
DCNN [291]	–	21.66	15.49	12.90	30.22	22.06	19.01	57.54	43.12	38.80	63.29	46.86	42.42
DEVIANT	–	**24.63**	**16.54**	**14.52**	**32.60**	**23.04**	**19.99**	**61.00**	**46.00**	**40.18**	**65.28**	**49.63**	**43.50**

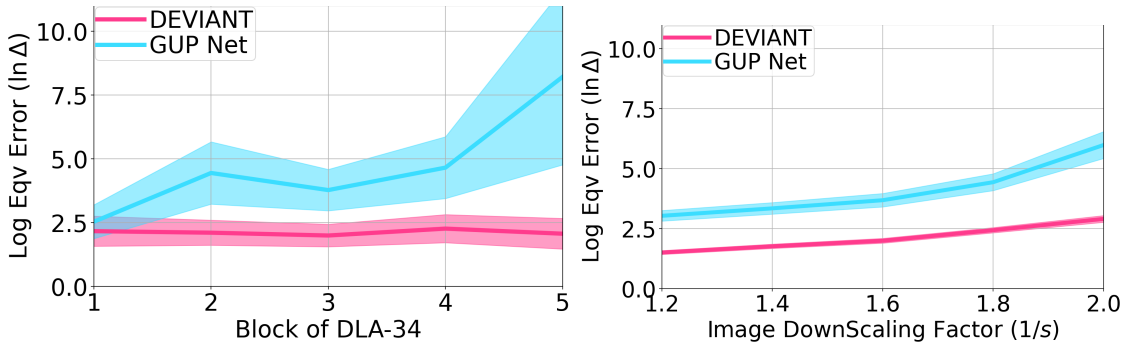
show that DCNN performs sub-par to DEVIANT. This is expected because dilation corresponds to integer scales [267] while the scaling is generally a float in monocular detection. D4LCN [52] uses monocular depth as input to adjust the receptive field. DEVIANT (without depth) also outperforms D4LCN on Hard cars, which are more distant.

(d) Other Convolutions. We now compare with other known convolutions in literature such as Log-polar convolution [313], Dilated convolution [291] convolution and DISCO [225] in Tab. 3.11. The results show that the log-polar convolution does not work well, and SES convolutions are better suited to embed depth (scale) equivariance. As described in Sec. 3.4, we investigate the behavior of log-polar convolution through a small experiment. We calculate the SSIM [258] of the original image and the image obtained after the upscaling, log-polar, inverse log-polar, and downscaling blocks. We then average the SSIM over all KITTI Val images. We repeat this experiment for multiple image heights and scaling factors. The ideal SSIM should have been one. However, Fig. 3.2c shows that SSIM does not reach 1 even after upscaling by 4. This result confirms that log-polar convolution loses information at low resolutions resulting in inaccurate detection.

Next, the results show that dilated convolution [291] performs sub-par to DEVIANT. Moreover, DISCO [225] also does not outperform SES convolution which agrees with the 2D tracking results of [225].

(e) Feature Pyramid Network (FPN). Our baseline GUP Net [159] uses FPN [138] and Tab. 3.5 shows that DEVIANT outperforms GUP Net. Hence, we conclude that equivariance also benefits models which use FPN.

Comparison of Equivariance Error. We next quantitatively evaluate the scale equivariance of DEVIANT vs. GUP Net [159], using the equivariance error metric [227]. The equivariance error



(a) At blocks (depths) of backbone.

(b) Varying scaling factors.

Figure 3.4 **Log Equivariance Error** (Δ) comparison for DEVIANT and GUP Net at **(a)** different blocks with random image scaling factors **(b)** different image scaling factors at depth 3. DEVIANT shows **lower** scale equivariance error than vanilla GUP Net [159].

Δ is the normalized difference between the scaled feature map and the feature map of the scaled image, and is given by $\Delta = \frac{1}{N}\sum_{i=1}^{N} \frac{||\mathcal{T}_{s_i}\Phi(h_i) - \Phi(\mathcal{T}_{s_i}h_i)||_2^2}{||\mathcal{T}_{s_i}\Phi(h_i)||_2^2}$, where Φ denotes the neural network, $\mathcal{T}_{s_i}$ is the scaling transformation for the image i, and N is the total number of images. The equivariance error is zero if the scale equivariance is perfect. We plot the log of this error at different blocks of DEVIANT and GUP Net backbones and also plot at different downscaling of KITTI Val images in Fig. 3.4. The plots show that DEVIANT has low equivariance error than GUP Net. This is expected since the feature maps of the proposed DEVIANT are additionally equivariant to scale transformation s (depth translations). We also visualize the equivariance error for a validation image and for the objects of this image in Figs. C.8a and C.8b in the supplementary. The qualitative plots also show a lower error for the proposed DEVIANT, which agrees with Fig. 3.4. Fig. C.8ba shows that equivariance error is particularly low for nearby cars which also justifies the good performance of DEVIANT on Easy (nearby) cars in Tabs. 3.3 and 3.5.

Does 2D Detection Suffer? We now investigate whether 2D detection suffers from using DEVIANT backbones in Tab. 3.10. The results show that DEVIANT introduces minimal decrease in the 2D detection performance. This is consistent with [226], who report that 2D tracking improves with the SE networks.

Table 3.10 **3D and 2D detection** on KITTI Val cars.

Method	IoU ≥ 0.7						IoU ≥ 0.5					
	AP $_{3D\|R_{40}}$ [%](↑)			AP $_{2D\|R_{40}}$ [%](↑)			AP $_{3D\|R_{40}}$ [%](↑)			AP $_{2D\|R_{40}}$ [%](↑)		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [159]	21.10	15.48	12.88	96.78	88.87	79.02	58.95	43.99	38.07	99.52	91.89	81.99
DEVIANT (Ours)	24.63	16.54	14.52	96.68	88.66	78.87	61.00	46.00	40.18	97.12	91.77	81.93

Table 3.11 **Ablation studies** on KITTI Val cars.

Change from DEVIANT:		IoU$_{3D}$ ≥ 0.7						IoU$_{3D}$ ≥ 0.5					
Changed	From ⟶To	AP $_{3D\|R_{40}}$ [%](↑)			AP $_{BEV\|R_{40}}$ [%](↑)			AP $_{3D\|R_{40}}$ [%](↑)			AP $_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Convolution	SES→Vanilla	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
	SES→Log-polar [313]	9.19	6.77	5.78	16.39	11.15	9.80	40.51	27.62	23.90	45.66	31.34	25.80
	SES→Dilated [291]	21.66	15.49	12.90	30.22	22.06	19.01	57.54	43.12	38.80	63.29	46.86	42.42
	SES→DISCO [225]	20.21	13.84	11.46	28.56	19.38	16.41	55.22	39.76	35.37	59.46	43.16	38.52
Downscale α	10% → 5%	24.24	16.51	14.43	31.94	22.86	19.82	60.64	44.46	40.02	64.68	49.30	43.49
	10% → 20%	22.19	15.85	13.48	31.15	23.01	19.90	61.24	44.93	40.22	67.46	50.10	43.83
BNP	SE→ Vanilla	24.39	16.20	14.36	32.43	22.53	19.70	62.81	46.14	40.38	67.87	50.23	44.08
Scales	3 → 1	23.20	16.29	13.63	31.76	23.23	19.97	61.90	46.66	40.61	67.37	50.31	43.93
	3 → 2	24.15	16.48	14.55	32.42	23.17	20.07	61.05	46.34	40.46	67.36	50.32	44.07
—	**DEVIANT (best)**	**24.63**	**16.54**	**14.52**	**32.60**	**23.04**	**19.99**	**61.00**	**46.00**	**40.18**	**65.28**	**49.63**	**43.50**

3.5.3 Ablation Studies on KITTI Val

Tab. 3.11 compares the modifications of our approach on KITTI Val cars based on the experimental settings of Sec. 3.5.

(a) Floating or Integer Downscaling? We next investigate the question that whether one should use floating or integer downscaling factors for DEVIANT. We vary the downscaling factors as $(1+2\alpha, 1+\alpha, 1)$ and therefore, our scaling factor $s = \left(\frac{1}{1+2\alpha}, \frac{1}{1+\alpha}, 1\right)$. We find that α of 10% works the best. We again bring up the dilated convolution (Dilated) results at this point because dilation is a scale equivariant operation for integer downscaling factors [267] ($\alpha = 100\%$, $s = 0.5$). Tab. 3.11 results suggest that the downscaling factors should be floating numbers.

(b) SE BNP. As described in Sec. 3.4, we ablate DEVIANT against the case when only convolutions are SE but BNP layers are not. So, we place Scale-Projection [227] immediately after every SES convolution. Tab. 3.11 shows that such a network performs slightly sub-optimal to our final model.

(c) Number of Scales. We next ablate against the usage of Hermite scales. Using three scales performs better than using only one scale especially on Mod and Hard objects, and slightly better than using two scales.

Table 3.12 **Waymo Val vehicles detection results**. [Key: **Best**, **Second Best**]

IoU_{3D}	Difficulty	Method	Extra	AP_{3D} [%](↑)				APH_{3D} [%](↑)			
				All	0-30	30-50	50-∞	All	0-30	30-50	50-∞
0.7	Level_1	CaDDN [199]	LiDAR	5.03	14.54	1.47	0.10	4.99	14.43	1.45	0.10
		PatchNet [162] in [246]	Depth	0.39	1.67	0.13	0.03	0.39	1.63	0.12	0.03
		PCT [246]	Depth	0.89	3.18	0.27	0.07	0.88	3.15	0.27	0.07
		M3D-RPN [15] in [199]	–	0.35	1.12	0.18	0.02	0.34	1.10	0.18	0.02
		GUP Net (Retrained) [159]	–	**2.28**	**6.15**	**0.81**	**0.03**	**2.27**	**6.11**	**0.80**	**0.03**
		DEVIANT (Ours)	–	**2.69**	**6.95**	**0.99**	**0.02**	**2.67**	**6.90**	**0.98**	**0.02**
0.7	Level_2	CaDDN [199]	LiDAR	4.49	14.50	1.42	0.09	4.45	14.38	1.41	0.09
		PatchNet [162] in [246]	Depth	0.38	1.67	0.13	0.03	0.36	1.63	0.11	0.03
		PCT [246]	Depth	0.66	3.18	0.27	0.07	0.66	3.15	0.26	0.07
		M3D-RPN [15] in [199]	–	0.33	1.12	0.18	**0.02**	0.33	1.10	0.17	**0.02**
		GUP Net (Retrained) [159]	–	**2.14**	**6.13**	**0.78**	**0.02**	**2.12**	**6.08**	**0.77**	**0.02**
		DEVIANT (Ours)	–	**2.52**	**6.93**	**0.95**	**0.02**	**2.50**	**6.87**	**0.94**	**0.02**
0.5	Level_1	CaDDN [199]	LiDAR	17.54	45.00	9.24	0.64	17.31	44.46	9.11	0.62
		PatchNet [162] in [246]	Depth	2.92	10.03	1.09	0.23	2.74	9.75	0.96	0.18
		PCT [246]	Depth	4.20	14.70	1.78	0.39	4.15	14.54	1.75	0.39
		M3D-RPN [15] in [199]	–	3.79	11.14	2.16	**0.26**	3.63	10.70	2.09	**0.21**
		GUP Net (Retrained) [159]	–	**10.02**	**24.78**	**4.84**	**0.22**	**9.94**	**24.59**	**4.78**	**0.22**
		DEVIANT (Ours)	–	**10.98**	**26.85**	**5.13**	0.18	**10.89**	**26.64**	**5.08**	0.18
0.5	Level_2	CaDDN [199]	LiDAR	16.51	44.87	8.99	0.58	16.28	44.33	8.86	0.55
		PatchNet [162] in [246]	Depth	2.42	10.01	1.07	0.22	2.28	9.73	0.97	0.16
		PCT [246]	Depth	4.03	14.67	1.74	0.36	4.15	14.51	1.71	0.35
		M3D-RPN [15] in [199]	–	3.61	11.12	2.12	**0.24**	3.46	10.67	2.04	**0.20**
		GUP Net (Retrained) [159]	–	**9.39**	**24.69**	**4.67**	**0.19**	**9.31**	**24.50**	**4.62**	**0.19**
		DEVIANT (Ours)	–	**10.29**	**26.75**	**4.95**	0.16	**10.20**	**26.54**	**4.90**	0.16

3.5.4 Waymo Val Mono3D

We also benchmark our method on the Waymo dataset [230] which has more variability than KITTI. Tab. 3.12 shows the results on Waymo Val split. The results show that DEVIANT outperforms the baseline GUP Net [159] on multiple levels and multiple thresholds. The biggest gains are on the nearby objects which is consistent with Tabs. 3.3 and 3.5. Interestingly, DEVIANT also outperforms PatchNet [162] and PCT [246] without using depth. Although the performance of DEVIANT lags CaDDN [199], it is important to stress that CaDDN uses LiDAR data in training, while DEVIANT is an image-only method.

3.6 Conclusions

This chapter studies the modeling error in monocular 3D detection in detail and takes the first step towards convolutions equivariant to arbitrary 3D translations in the projective manifold. Since the depth is the hardest to estimate for this task, this chapter proposes Depth Equivariant

Network (DEVIANT) built with existing scale equivariant steerable blocks. As a result, DEVIANT is equivariant to the depth translations in the projective manifold whereas vanilla networks are not. The additional depth equivariance forces the DEVIANT to learn consistent depth estimates and therefore, DEVIANT achieves SoTA detection results on KITTI and Waymo datasets in the image-only category and performs competitively to methods using extra information. Moreover, DEVIANT works better than vanilla networks in cross-dataset evaluation. Future works include applying the idea to Pseudo-LiDAR [254], and monocular 3D tracking.

Limitation. DEVIANT does not model 3D equivariance but only a special case of 3D equivariance. Considerably less number of boxes are detected in the cross-dataset evaluation.

CHAPTER 4

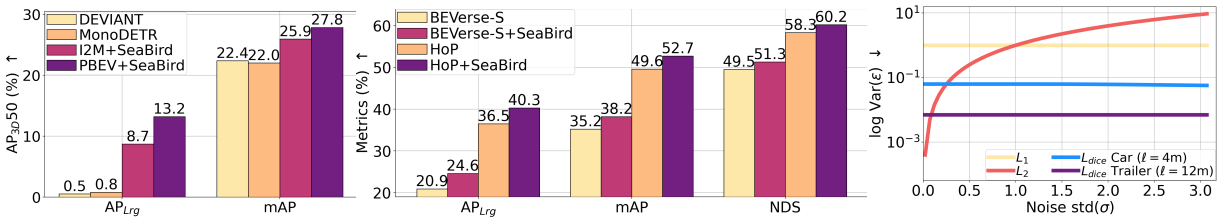
SEABIRD: SEGMENTATION IN BIRD'S VIEW WITH DICE LOSS IMPROVES MONOCULAR 3D DETECTION OF LARGE OBJECTS

Monocular 3D detectors achieve remarkable performance on cars and smaller objects. However, their performance drops on larger objects, leading to fatal accidents. Some attribute the failures to training data scarcity or their receptive field requirements of large objects. In this chapter, we highlight this understudied problem of generalization to large objects. We find that modern frontal detectors struggle to generalize to large objects even on nearly balanced datasets. We argue that the cause of failure is the sensitivity of depth regression losses to noise of larger objects. To bridge this gap, we comprehensively investigate regression and dice losses, examining their robustness under varying error levels and object sizes. We mathematically prove that the dice loss leads to superior noise-robustness and model convergence for large objects compared to regression losses for a simplified case. Leveraging our theoretical insights, we propose SeaBird (Segmentation in Bird's View) as the first step towards generalizing to large objects. SeaBird effectively integrates BEV segmentation on foreground objects for 3D detection, with the segmentation head trained with the dice loss. SeaBird achieves SoTA results on the KITTI-360 leaderboard and improves existing detectors on the nuScenes leaderboard, particularly for large objects.

4.1 Introduction

Monocular 3D object detection (Mono3D) task aims to estimate both the 3D position and dimensions of objects in a scene from a single image. Its applications span autonomous driving [108, 132, 181], robotics [213], and augmented reality [2, 172, 183, 293], where accurate 3D understanding of the environment is crucial. Our study focuses explicitly on 3D object detectors applied to autonomous vehicles (AVs), considering the challenges and motivations deviate drastically across different applications.

AVs demand object detectors that generalize to diverse intrinsics [14], camera-rigs [94, 104], rotations [177], weather and geographical conditions [54] and also are robust to adversarial examples [310]. Since each of these poses a significant challenge, recent works focus exclusively on the



(a) Improve KITTI-360 SoTA. (b) Improve nuScenes Val SoTA. (c) Theory Advancement.

Figure 4.1 **Teaser (a)** SoTA frontal detectors struggle with large objects (low AP_{Lrg}) even on a nearly balanced KITTI-360 dataset. Our proposed SeaBird achieves significant Mono3D improvements, particularly for large objects. **(b)** SeaBird also improves two SoTA BEV detectors, BEVerse-S [303] and HoP [311] on the nuScenes dataset, particularly for large objects. **(c)** Plot of convergence variance $\text{Var}(\epsilon)$ of dice and regression losses with the noise σ in depth prediction. The y-axis denotes the deviation from the optimal weight, so the lower the better. SeaBird leverages **dice loss**, which we prove is more noise-robust than regression losses for large objects.

generalization of object detectors to all these out-of-distribution shifts. However, our focus is on the generalization of another type, which, thus far, has been understudied in the literature – *Mono3D generalization to large objects*.

Large objects like trailers, buses and trucks are harder to detect [268] in Mono3D, sometimes resulting in fatal accidents [23, 60]. Some attribute these failures to training data scarcity [308] or the receptive field requirements [268] of large objects, but, to the best of our knowledge, no existing literature provides a comprehensive analytical explanation for this phenomenon. The goal of this chapter is, thus, to bring understanding and a first analytical approach to this real-world problem in the AV space – Mono3D generalization to large objects.

We conjecture that the generalization issue stems not only from limited training data or larger receptive field but also from the noise sensitivity of depth regression losses in Mono3D. To substantiate our argument, we analyze the Mono3D performance of state-of-the-art (SoTA) frontal detectors on the KITTI-360 dataset [136], which includes almost equal number ($1:2$) of large objects and cars. We observe that SoTA detectors struggle with large objects on this dataset (Fig. 4.1a). Next, we carefully investigate the SGD convergence of losses used in Mono3D task and mathematically prove that the dice loss, widely used in BEV segmentation, exhibits superior noise-robustness than the regression losses, particularly for large objects (Fig. 4.1c). Thus, the dice loss facilitates better model convergence than regression losses, improving Mono3D of large

objects.

Incorporating dice loss in detection introduces unique challenges. Firstly, the dice loss does not apply to sparse detection centers and only incorporates depth information when used in the BEV space. Secondly, naive joint training of Mono3D and BEV segmentation tasks with image inputs does not always benefit Mono3D task [132, 167] due to negative transfer [45], and the underlying reasons remain unclear. Fortunately, many Mono3D segmentors and detectors are in the BEV space, where the BEV segmentor can seamlessly apply dice loss and the BEV detector can readily benefit from the segmentor in the same space. To mitigate negative transfer, we find it effective to train the BEV segmentation head on the foreground detection categories.

Building upon our theoretical findings about the dice loss, we propose a simple and effective pipeline called Segmentation in Bird's View (SeaBird) for enhancing Mono3D of large objects. SeaBird employs a sequential approach for the BEV segmentation and Mono3D heads (Fig. 4.2). SeaBird first utilizes a BEV segmentation head to predict the segmentation of only foreground objects, supervised by the dice loss. The dice loss offers superior noise-robustness for large objects, ensuring stable convergence, while focusing on foreground objects in segmentation mitigates negative transfer. Subsequently, SeaBird concatenates the resulting BEV segmentation map with the original BEV features as an additional feature channel and feeds this concatenated feature to a Mono3D head supervised by Mono3D losses[1]. Building upon this, we adopt a two-stage training pipeline: the first stage exclusively focuses on training the BEV segmentation head with dice loss, which fully exploits its noise-robustness and superior convergence in localizing large objects. The second stage involves both the detection loss and dice loss to finetune the Mono3D head.

In our experiments, we first comprehensively evaluate SeaBird and conduct ablations on the balanced single-camera KITTI-360 dataset [136]. SeaBird outperforms the SoTA baselines by a substantial margin. Subsequently, we integrate SeaBird as a plug-in-and-play module into two SoTA detectors on the multi-camera nuScenes dataset [22]. SeaBird again significantly improves the original detectors, particularly on large objects. Additionally, SeaBird consistently enhances

[1]Only Mono3D head predicts additional 3D attributes, namely object's height and elevation.

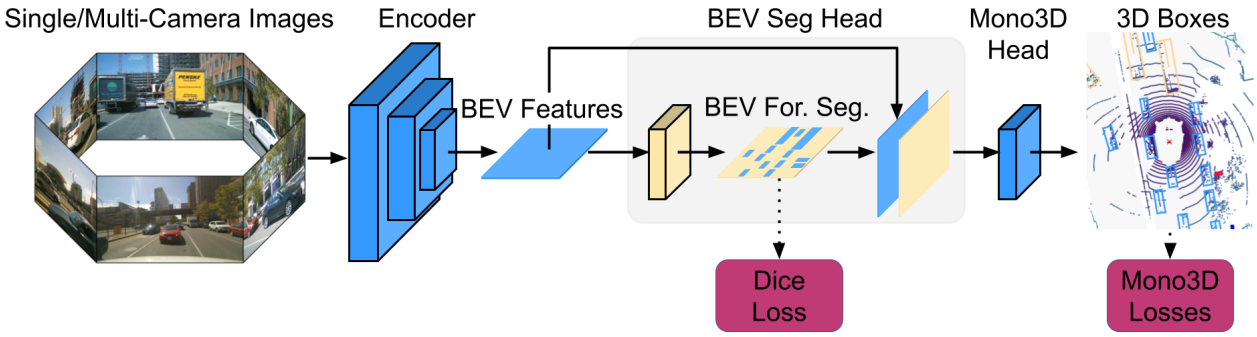


Figure 4.2 **SeaBird Pipeline.** SeaBird uses the predicted BEV foreground segmentation (For. Seg.) map to predict accurate 3D boxes for large objects. SeaBird training protocol involves BEV segmentation pre-training with the noise-robust dice loss and Mono3D fine-tuning.

Mono3D performance across backbones with those two SoTA detectors (Fig. 4.1b), demonstrating its utility in both edge and cloud deployments.

In summary, we make the following contributions:

- We highlight the understudied problem of generalization to large objects in Mono3D, showing that even on nearly balanced datasets, SoTA frontal models struggle to generalize due to the noise sensitivity of regression losses.
- We mathematically prove that the dice loss leads to superior noise-robustness and model convergence for large objects compared to regression losses for a simplified case and provide empirical support for more general settings.
- We propose SeaBird, which treats BEV segmentation head on foreground objects and Mono3D head sequentially and trains in a two-stage protocol to fully harness the noise-robustness of the dice loss.
- We empirically validate our theoretical findings and show significant improvements, particularly for large objects, on both KITTI-360 and nuScenes leaderboards.

4.2 Related Works

Mono3D. Mono3D popularity stems from its high accessibility from consumer vehicles compared to LiDAR/Radar-based detectors [155, 215, 290] and computational efficiency compared to stereo-based detectors [34]. Earlier approaches [31, 186] leverage hand-crafted features, while the recent ones use deep networks. Advancements include introducing new architectures [89, 217, 275],

equivariance [29,108], losses [15,35], uncertainty [111,159] and incorporating auxiliary tasks such as depth [175, 301], NMS [109, 147, 216], corrected extrinsics [307], CAD models [25, 117, 154] or LiDAR [199] in training. A particular line of work called Pseudo-LiDAR [165, 254] shows generalization by first estimating the depth, followed by a point cloud-based 3D detector.

Another line of work encodes image into latent BEV features [164] and attaches multiple heads for downstream tasks [303]. Some focus on pre-training [272] and rotation-equivariant convolutions [59]. Others introduce new coordinate systems [95], queries [128, 161], or positional encoding [219] in a transformer-based detection framework [24]. Some use pixel-wise depth [88], object-wise depth [38, 40, 141], or depth-aware queries [296], while many utilize temporal fusion [17, 150, 248, 261] to boost performance. A few use longer frame history [182, 311], distillation [105, 260] or stereo [125, 261]. We refer to [163, 167] for the survey. SeaBird also builds upon the BEV-based framework since it flexibly accepts single or multiple images as input and uses dice loss. Different from the majority of other detectors, SeaBird improves Mono3D of large objects using the power of dice loss. SeaBird is also the first work to mathematically prove and justify this loss choice for large objects.

BEV Segmentation. BEV segmentation typically utilizes BEV features transformed from 2D image features. Various methods encode single or multiple images into BEV features using MLPs [180] or transformers [205, 211]. Some employ learned depth distribution [83, 188], while others use attention [211, 305] or attention fields [37]. Image2Maps [211] utilizes polar ray, while PanopticBEV [72] uses transformers. FIERY [83] introduces uncertainty modelling and temporal fusion, while Simple-BEV [74] uses radar aggregation. Since BEV segmentation lacks object height and elevation, one also needs a Mono3D head to predict 3D boxes.

Joint Mono3D and BEV Segmentation. Joint 3D detection and BEV segmentation using LiDAR data [58, 215] as input benefits both tasks [252, 281]. However, joint learning on image data often hinders detection performance [132, 167, 272, 303], while the BEV segmentation improvement is inconsistent across categories [167]. Unlike these works which treat the two heads in parallel and decrease Mono3D performance [167], SeaBird treats the heads sequentially and increases Mono3D

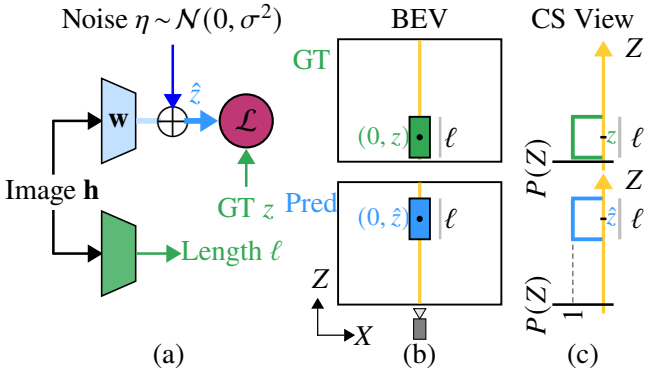


Figure 4.3 **(a) Problem setup**. The single-layer neural network takes an image **h** (or its features) and predicts depth $\hat{z}$ and the object length ℓ. The noise η is the additive error in depth prediction and is a normal random variable. The GT depth z supervises the predicted depth $\hat{z}$ with a loss $\mathcal{L}$ in training. We assume the network predicts the GT length ℓ. Frontal detectors directly regress the depth with $\mathcal{L}_1$, $\mathcal{L}_2$, or Smooth $\mathcal{L}_1$ loss, while SeaBird projects to BEV plane and supervises through dice loss $\mathcal{L}_{dice}$. **(b) Shifting of predictions (blue)** in BEV along the ray due to the noise η. **(c) Cross Section (CS) view** along the ray with classification scores $P(Z)$.

performance, particularly for large objects.

4.3 SeaBird

SeaBird is driven by a deep understanding of the distinctions between monocular regression and BEV segmentation losses. Thus, in this section, we delve into the problem and discuss existing results. We then present our theoretical findings and, subsequently, introduce our pipeline.

We introduce the problem and refer to Lemma 1 from the literature [113, 214], which *evaluates* loss quality by measuring the deviation of trained weight (after SGD updates) from the optimal weight. Fig. 4.3a illustrates the problem setup. Figs. 4.3b and 4.3c visualize the BEV and cross-section view, respectively. Since this deviation depends on the gradient variance of losses, we next derive the gradient variance of the dice loss in Lemma 2. By comparing the distance between trained weight and optimal weight, we assess the effectiveness of dice loss versus MAE ($\mathcal{L}_1$) and MSE ($\mathcal{L}_2$) losses in Lemma 3, and choose the representation and loss combination. Combining these findings, we establish Th. 2 that the model trained with dice loss achieves better AP than the model trained with regression losses. Finally, we present our pipeline, SeaBird, which integrates BEV segmentation supervised by dice loss for Mono3D.

4.3.1 Background and Problem Statement

Mono3D networks [108, 159] commonly employ regression losses, such as $\mathcal{L}_1$ or $\mathcal{L}_2$ loss, to compare the predicted depth with ground truth (GT) depth [108, 303]. In contrast, BEV segmentation utilizes dice loss [211] or cross-entropy loss [83] at each BEV location, comparing it with GT. Despite these distinct loss functions, we evaluate their effectiveness under an idealized model, where we measure the model *quality* by the expected deviation of trained weight (after SGD updates) from the optimal weight [214].

Lemma 1. ***Convergence analysis*** *[214]. Consider a linear regression model with trainable weight* $\mathbf{w}$ *for depth prediction* $\hat{z}$ *from an image* $\mathbf{h}$*. Assume the noise* η *is an additive error in depth prediction and is a normal random variable* $\mathcal{N}(0, \sigma^2)$*. Also, assume SGD optimizes the model parameters with loss function* $\mathcal{L}$ *during training with square summable steps* s_j*, i.e.* $s = \lim_{t\to\infty} \sum_{j=1}^{t} s_j^2$ *exists and* η *is independent of the image. Then, the expected deviation of the trained weight* ${}^{\mathcal{L}}\mathbf{w}_\infty$ *from the optimal weight* $\mathbf{w}_*$ *obeys*

$$\mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) = c_1 Var(\epsilon) + c_2, \tag{4.1}$$

where $\epsilon = \frac{\partial \mathcal{L}(\eta)}{\partial \eta}$ *is the gradient of the loss* $\mathcal{L}$ *wrt noise,* $c_1 = s\mathbb{E}(\mathbf{h}^T\mathbf{h})$ *and* c_2 *are constants independent of the loss.*

We refer to Sec. D.1.1 for the proof. Eq. (4.1) demonstrates that training losses $\mathcal{L}$ exhibit varying gradient variances $\text{Var}(\epsilon)$. Hence, comparing this term for different losses allows us to evaluate their quality.

4.3.2 Loss Analysis: Dice vs. Regression

Given that [214] provides the gradient variance $\text{Var}(\epsilon)$, for $\mathcal{L}_1$ and $\mathcal{L}_2$ losses, we derive the corresponding gradient variance for dice and IoU losses in this chapter to facilitate comparison. First, we express the dice loss, $\mathcal{L}_{dice}$, as a function of noise η as per its definition from [211] for Fig. 4.3c as:

$$\mathcal{L}_{dice}(\eta) = 1 - 2\frac{\text{Pred GT}}{\text{Pred} + \text{GT}} = \begin{cases} 1 - 2\frac{\ell - |\eta|}{2\ell}, & |\eta| \le \ell \\ 1, & |\eta| \ge \ell \end{cases}$$

Table 4.1 **Convergence variance** of training loss functions. Gradient variance of $\mathcal{L}_{dice}$ is more noise-robust for large objects, resulting in better detectors. We do not analyze cross-entropy loss theoretically since its $\mathrm{Var}(\epsilon)$ is infinite, but empirically in Tab. 4.5.

Loss $\mathcal{L}$	Gradient ϵ	$\mathrm{Var}(\epsilon)$ ($\downarrow$)
$\mathcal{L}_1$ [214] (App. D.1.2.1)	$\mathrm{sgn}(\eta)$	1
$\mathcal{L}_2$ [214] (App. D.1.2.2)	η	σ^2
Dice (Lemma 2)	$\begin{cases} \frac{\mathrm{sgn}(\eta)}{\ell} & , \lvert\eta\rvert \le \ell \\ 0 & , \lvert\eta\rvert \ge \ell \end{cases}$	$\frac{1}{\ell^2}\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)$

$$\implies \mathcal{L}_{dice}(\eta) = \begin{cases} \frac{|\eta|}{\ell} \, , |\eta| \le \ell \\ \\ 1 \quad , |\eta| \ge \ell \end{cases} , \tag{4.2}$$

where ℓ denotes the object length. Eq. (4.2) shows that the dice loss $\mathcal{L}_{dice}$ depends on the object size ℓ. With the given dice loss $\mathcal{L}_{dice}$, we proceed to derive the following lemma:

Lemma 2. ***Gradient variance of dice loss.*** *Let $\eta = \mathcal{N}(0, \sigma^2)$ be an additive normal random variable and ℓ be the object length. Let Erf be the error function. Then, the gradient variance of the dice loss $Var_{dice}(\epsilon)$ wrt noise η is*

$$Var_{dice}(\epsilon) = \frac{1}{\ell^2} Erf\left(\frac{\ell}{\sqrt{2}\sigma}\right). \tag{4.3}$$

We refer to Sec. D.1.2.3 for the proof. Eq. (4.3) shows that gradient variance of the dice loss $\mathrm{Var}_{dice}(\epsilon)$ also varies inversely to the object size ℓ and the noise deviation σ (See Sec. D.1.5). These two properties of dice loss are particularly beneficial for large objects.

Tab. 4.1 summarizes these losses, their gradients, and gradient variances. With $\mathrm{Var}_{dice}(\epsilon)$ derived for the dice loss, we now compare the deviation of trained weight with the deviations from $\mathcal{L}_1$ or $\mathcal{L}_2$ losses, leading to our next lemma.

Lemma 3. ***Dice model is closer to optimal weight than regression loss models.*** *Based on Lemma 1 and assuming the object length ℓ is a constant, if σ_m is the solution of the equation $\sigma^2 = \frac{1}{\ell^2} Erf\left(\frac{\ell}{\sqrt{2}\sigma}\right)$ and the noise deviation $\sigma \ge \sigma_c = \max\left(\sigma_m, \frac{\sqrt{2}}{\ell} Erf^{-1}(\ell^2)\right)$, then the converged weight ${}^d\mathbf{w}_\infty$ with the dice loss $\mathcal{L}_{dice}$ is better than the converged weight ${}^r\mathbf{w}_\infty$ with the $\mathcal{L}_1$ or $\mathcal{L}_2$ loss, i.e.*

$$\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2\right) \le \mathbb{E}\left(\|{}^r\mathbf{w}_\infty - \mathbf{w}_*\|_2\right). \tag{4.4}$$

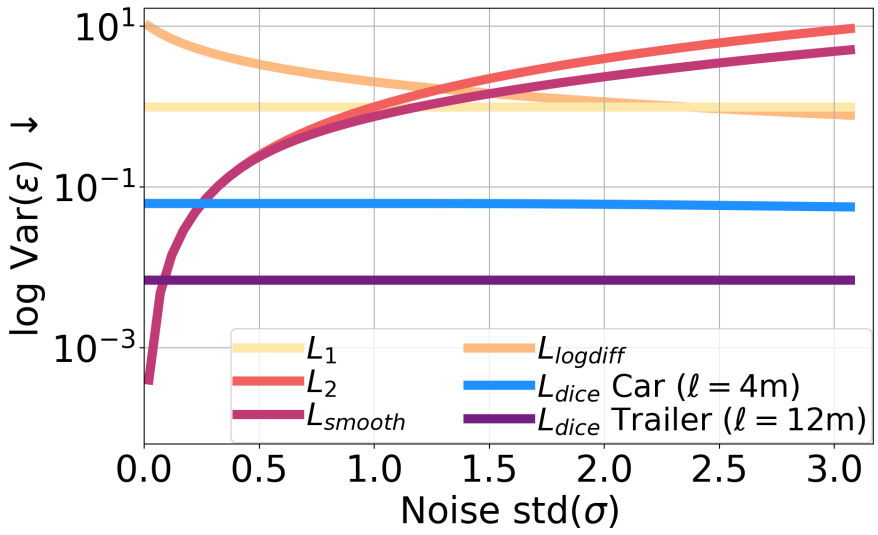


Figure 4.4 **Plot of convergence variance** Var(ϵ) of loss functions with the noise σ. Dice loss has minimum convergence variance with large noise, resulting in better detectors for large objects.

We refer to Sec. D.1.3 for the proof. Beyond noise deviation threshold $\sigma_c = \max\left(\sigma_m, \frac{\sqrt{2}}{\ell}\text{Erf}^{-1}(\ell^2)\right)$, the convergence gap between dice and regression losses widens as the object size ℓ increases. Fig. 4.4 depicts the superior convergence of dice loss compared to regression losses under increasing noise deviation σ pictorially. Taking the car category with $\ell = 4m$ and the trailer category with $\ell = 12m$ as examples, the noise threshold σ_c, beyond which dice loss exhibits better convergence, are $\sigma_c = 0.3m$ and $\sigma_c = 0.1m$ respectively. Combining these lemmas, we finally derive:

Theorem 2. ***Dice model has better AP_{3D}.*** *Assume the object length ℓ is a constant and depth is the only source of error for detection. Based on Lemma 1, if σ_m is the solution of the equation $\sigma^2 = \frac{1}{\ell^2} Erf\left(\frac{\ell}{\sqrt{2}\sigma}\right)$ and the noise deviation $\sigma \geq \sigma_c = \max\left(\sigma_m, \frac{\sqrt{2}}{\ell} Erf^{-1}(\ell^2)\right)$, then the Average Precision (AP_{3D}) of the dice model is better than AP_{3D} from $\mathcal{L}_1$ or $\mathcal{L}_2$ model.*

We refer to Sec. D.1.4 and Tab. D.1 for the proof and assumption comparisons respectively.

4.3.3 Discussions

Comparing classification and regression losses. We now explain how we compare classification (dice) and regression losses. Our analysis assumes one-class classification in BEV segmentation with perfect predicted foreground scores $P(Z) = 1$ (Fig. 4.3c). Hence, dice analysis focuses on object localization along the BEV ray (Fig. 4.3b) instead of classification probabilities thus allowing comparison of dice and regression losses. Lemma 1 links these losses by comparing the deviation

of learned and optimal weights.

Regression losses work better than dice loss for regression tasks? Our key message is NOT always! We mathematically and empirically show that regression losses work better only when the **noise σ is less** in Fig. 4.4.

4.3.4 SeaBird Pipeline

Architecture. Based on theoretical insights of Th. 2, we propose SeaBird, a novel pipeline, in Fig. 4.2. To effectively involve the dice loss which originally designed for segmentation task to assist Mono3D, SeaBird treats BEV segmentation of foreground objects and Mono3D head sequentially. Although BEV segmentation map provides depth information (hardest [108, 166] Mono3D parameter), it lacks elevation and height information for Mono3D task. To address this, SeaBird concatenates BEV features with predicted BEV segmentation (Fig. 4.2), and feeds them into the detection head to predict 3D boxes in a 7-DoF representation: BEV 2D position, elevation, 3D dimension, and yaw. Unlike most works [132, 303] that treat segmentation and detection branches in parallel, the sequential design directly utilizes refined BEV localization information to enhance Mono3D. Ablations in Sec. 4.4.2 validate this design choice. We defer the details of baselines to Sec. 4.4. Notably, our foreground BEV segmentation supervision with dice loss does not require dense BEV segmentation maps, as we efficiently prepare them from GT 3D boxes.

Training Protocol. SeaBird trains the BEV segmentation head first, employing the dice loss between the predicted and the GT BEV semantic segmentation maps, which fully utilizes the dice loss's noise-robustness and superior convergence in localizing large objects. In the second stage, we jointly fine-tune the BEV segmentation head and the Mono3D head. We validate the effectiveness of training protocol via the ablation in Sec. 4.4.2.

4.4 Experiments

Datasets. Our experiments utilize two datasets with large objects: KITTI-360 [136] and nuScenes [22] encompassing both single-camera and multi-camera configurations. We opt for KITTI-360 instead of KITTI [67] for four reasons: 1) KITTI-360 includes large objects, while KITTI does not; 2) KITTI-360 exhibits a balanced distribution of large objects and cars; 3) an extended version,

Table 4.2 **Datasets comparison.** We use KITTI-360 and nuScenes datasets for our experiments. See Fig. D.2 for the skewness.

	KITTI [67]	Waymo [230]	KITTI-360 [136]	nuScenes [22]
Large objects	×	×	✓	✓
Balanced	×	×	✓	×
BEV Seg. GT	×	✓	✓	✓
#images (k)	4	52 [108]	49	168

KITTI-360 PanopticBEV [72], includes BEV segmentation GT for ablation studies, while KITTI 3D detection and the Semantic KITTI dataset [6] do not overlap in sequences; 4) KITTI-360 contains about 10× more images than KITTI. We compare these datasets in Tab. 4.2 and show their skewness in Fig. D.2.

Data Splits. We use the following splits of the two datasets:

- *KITTI-360 Test split*: This benchmark [136] contains 300 training and 42 testing windows. These windows contain 61,056 training and 910 testing images.
- *KITTI-360 Val split*: It partitions the official train into 239 train and 61 validation windows [136]. This split contains 48,648 training and 1,294 validation images.
- *nuScenes Test split*: It has 34,149 training and 6,006 testing samples [22] from the six cameras. This split contains 204,894 training and 36,036 testing images.
- *nuScenes Val split*: It has 28,130 training and 6,019 validation samples [22] from the six cameras. This split contains 168,780 training and 36,114 validation images.

Evaluation Metrics. We use the following metrics:

- *Detection*: KITTI-360 uses the mean AP $_{3D}$ 50 percentage across categories to benchmark models [136]. nuScenes [22] uses the nuScenes Detection Score (NDS) as the metric. NDS is the weighted average of mean AP (mAP) and five TP metrics. We also report mAP over large categories (truck, bus, trailers and construction vehicles), cars, and small categories (pedestrians, motorcyle, bicycle, cone and barrier) as AP_{Lrg}, AP_{Car} and AP_{Sml} respectively.
- *Semantic Segmentation*: We report mean IoU over foreground and all categories at 200×200 resolution [211, 303].

KITTI-360 Baselines and SeaBird Implementation. Our evaluation on the KITTI-360 focuses on

the detectors taking single-camera image as input. We evaluate SeaBird pipelines against six SoTA frontal detectors: GrooMeD-NMS [109], MonoDLE [166], GUP Net [159], DEVIANT [108], Cube R-CNN [14] and MonoDETR [300]. The choice of these models encompasses anchor [14, 109] and anchor-free methods [108, 166], CNN [159, 166], group CNN [108] and transformer-based [300] architectures. Further, MonoDLE normalizes loss with GT box dimensions.

Due to SeaBird's BEV-based approach, we do not integrate it with these frontal view detectors. Instead, we extend two SoTA image-to-BEV segmentation methods, Image2Maps (I2M) [211] and PanopticBEV (PBEV) [72] with SeaBird. Since both BEV segmentors already include their own implementations of the image encoder, the image-to-BEV transform, and the segmentation head, implementing the SeaBird pipeline only involves adding a detection head, which we chose to be Box Net [289]. SeaBird extensions employ dice loss for BEV segmentation, Smooth $\mathcal{L}_1$ losses [69] in the BEV space to supervise the BEV 2D position, elevation, and 3D dimension, and cross entropy loss to supervise orientation.

nuScenes Baselines and SeaBird Implementation. We integrate SeaBird into two prototypical BEV-based detectors, BEVerse [303] and HoP [311] to prove the effectiveness of SeaBird. Our choice of these models encompasses both transformer and convolutional backbones, multi-head and single-head architectures, shorter and longer frame history, and non-query and query-based detectors. This comprehensively allows us to assess SeaBird's impact on large object detection. BEVerse employs a multi-head architecture with a transformer backbone and shorter frame history. HoP is single-head query-based SoTA model utilizing BEVDet4D [87] with CNN backbone, and longer frame history.

BEVerse [303] includes its own implementation of detection head and BEV segmentation head in parallel. We reorganize the two heads to follow our sequential design and adhere to our training protocol for network training. Since HoP [311] lacks a BEV segmentation head, we incorporate the one from BEVerse into this HoP extension with SeaBird.

Table 4.3 **KITTI-360 Test detection results.** SeaBird pipelines outperform all monocular baselines, and also outperform old LiDAR baselines. Click for the KITTI-360 leaderboard as well as our PBEV+SeaBird and I2M+SeaBird entries. [Key: **Best**, **Second Best**, L= LiDAR, C= Camera, †= Retrained].

Modality L	Modality C	Method	Venue	AP_{3D} 50 (↑) mAP [%]	AP_{3D} 25 (↑) mAP [%]
✓		L-VoteNet [194]	ICCV19	3.40	30.61
✓		L-BoxNet [194]	ICCV19	4.08	23.59
	✓	GrooMeD† [109]	CVPR21	0.17	16.12
	✓	MonoDLE† [166]	CVPR21	0.85	28.99
	✓	GUP Net† [159]	ICCV21	0.87	27.25
	✓	DEVIANT† [108]	ECCV22	0.88	26.96
	✓	Cube R-CNN† [14]	CVPR23	0.80	15.57
	✓	MonoDETR† [300]	ICCV23	0.79	27.13
	✓	**I2M+SeaBird**	CVPR24	**3.14**	**35.04**
	✓	**PBEV+SeaBird**	CVPR24	**4.64**	**37.12**

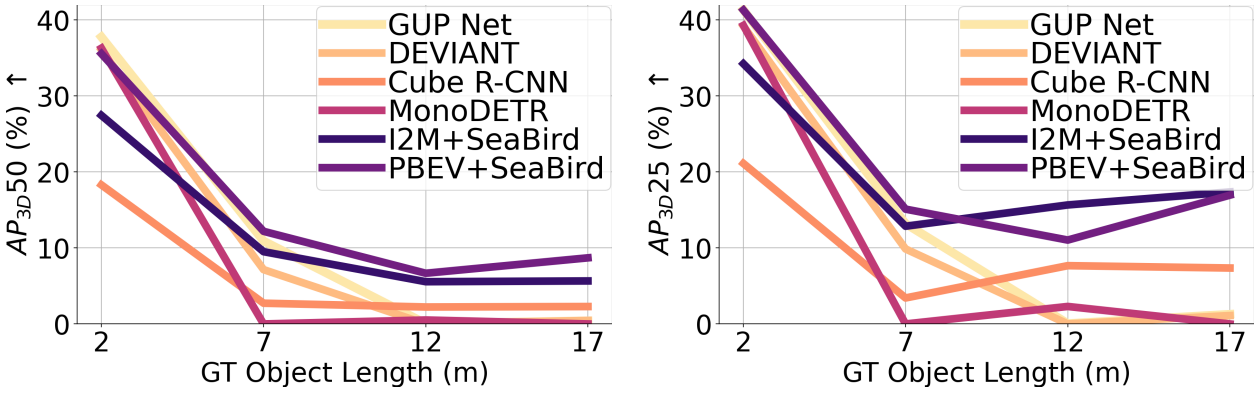
4.4.1 KITTI-360 Mono3D

KITTI-360 Test. Tab. 4.3 presents KITTI-360 leaderboard results, demonstrating the superior performance of both SeaBird pipelines compared to all monocular baselines across all metrics. Moreover, PBEV+SeaBird also outperforms both legacy LiDAR baselines on all metrics, while I2M+SeaBird surpasses them on the AP_{3D} 25 metric.

KITTI-360 Val. Tab. 4.4 presents the results on KITTI-360 Val split, reporting the **median** model over three different seeds with the model being the final checkpoint as [108]. SeaBird pipelines outperform all monocular baselines on all but one metric, similar to Tab. 4.3 results. Due to the dice loss in SeaBird, the biggest improvement shows up on larger objects. Tab. 4.4 also includes the upper-bound oracle, where we train the Box Net with the GT BEV segmentation maps.

Lengthwise AP Analysis. Th. 2 states that training a model with dice loss should lead to lower errors and, consequently, a better detector for large objects. To validate this claim, we analyze the detection performance with AP_{3D} 50 and AP_{3D} 25 metrics against the object's lengths. For this analysis, we divide objects into four bins based on their GT object length (max of sizes): $[0, 5)$, $[5, 10)$, $[10, 15)$, $[15 + m$. Fig. 4.5 shows that SeaBird pipelines excel for large objects, where the baselines' performance drops significantly.

BEV Semantic Segmentation. Tab. 4.4 also presents the BEV semantic segmentation results on the KITTI-360 Val split. SeaBird pipelines outperforms the baseline I2M [211], and achieve



(a) AP_{3D} 50 comparison.

(b) AP_{3D} 25 comparison.

Figure 4.5 **Lengthwise AP Analysis** of four SoTA detectors and two SeaBird pipelines on KITTI-360 Val split. SeaBird pipelines outperform all baselines on large objects with over 10m in length.

Table 4.4 **KITTI-360 Val detection and segmentation results.** SeaBird pipelines outperform all frontal monocular baselines, particularly for large objects. Dice loss in SeaBird also improves the BEV only (w/o dice) version of SeaBird pipelines. I2M and PBEV are BEV segmentors. So, we do not report their Mono3D performance. [Key: **Best**, **Second Best**, †= Retrained]

View	Method	BEV Seg Loss	Venue	AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)			BEV Seg IoU [%](↑)		
				AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
Frontal	GrooMeD-NMS† [109]	–	CVPR21	0.00	33.04	16.52	0.00	38.21	19.11	–	–	–
	MonoDLE† [166]		CVPR21	0.94	44.81	22.88	4.64	50.52	27.58	–	–	–
	GUP Net† [159]		ICCV21	0.54	**45.11**	22.83	0.98	50.52	25.75	–	–	–
	DEVIANT† [108]		ECCV22	0.53	44.25	22.39	1.01	48.57	24.79	–	–	–
	Cube R-CNN† [14]		CVPR23	0.75	22.52	11.63	5.55	27.12	16.34	–	–	–
	MonoDETR† [300]		ICCV23	0.81	43.24	22.02	4.50	48.69	26.60	–	–	–
BEV	I2M† [211]	Dice	ICRA22	–	–	–	–	–	–	20.46	38.04	29.25
	I2M+SeaBird	×	CVPR24	4.86	45.09	24.98	26.33	52.31	39.32	0.00	7.07	3.54
	I2M+SeaBird	Dice	CVPR24	**8.71**	43.19	25.95	**35.76**	52.22	**43.99**	23.23	39.61	31.42
	PBEV† [72]	CE	RAL22	–	–	–	–	–	–	**23.83**	**48.54**	**36.18**
	PBEV+SeaBird	×	CVPR24	7.64	**45.37**	**26.51**	29.72	**53.86**	41.79	2.07	1.47	1.57
	PBEV+SeaBird	Dice	CVPR24	**13.22**	42.46	**27.84**	**37.15**	**52.53**	**44.84**	**24.30**	**48.04**	**36.17**
	Oracle (GT BEV)		–	26.77	51.79	39.28	49.74	56.62	53.18	100.00	100.00	100.00

similar performance to PBEV [72] in BEV segmentation. We retrain all BEV segmentation models only on foreground detection categories for a fair comparison.

4.4.2 Ablation Studies on KITTI-360 Val

Tab. 4.5 ablates I2M [211] +SeaBird on the KITTI-360 Val split, following the experimental settings of Sec. 4.4.1.

Dice Loss. Tab. 4.5 shows that both dice loss and BEV representation are crucial to Mono3D of large objects. Replacing dice loss with MSE or Smooth $\mathcal{L}_1$ loss, or only BEV representation (w/o dice) reduces Mono3D performance.

Mono3D and BEV Segmentation. Tab. 4.5 shows that removing the segmentation head hinders

Table 4.5 **Ablation studies** on KITTI-360 Val. [Key: **Best**, **Second Best**]

Changed	From ➔ To	AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)			BEV Seg IoU [%](↑)			
		AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}	M_{All}
Segmentation Loss	Dice ➔ No Loss	4.86	**45.09**	24.98	26.33	52.31	39.32	0.00	7.07	3.54	–
	Dice ➔ Smooth $\mathcal{L}_1$	7.63	36.69	22.16	31.01	47.51	39.26	17.16	34.67	25.92	–
	Dice ➔ MSE	7.04	35.59	21.32	30.90	44.71	37.81	17.46	34.85	26.16	–
	Dice ➔ CE	7.06	35.60	21.33	**33.22**	47.60	40.41	21.83	38.11	29.97	–
Segmentation Head	Yes ➔ No	7.52	39.24	23.38	31.83	47.88	39.86	–	–	–	–
Detection Head	Yes ➔ No	–	–	–	–	–	–	20.46	38.04	29.25	–
Semantic Category	For. ➔ All	1.61	**44.12**	22.87	15.36	51.76	33.56	19.26	34.46	26.86	**24.34**
	For. ➔ Car	4.17	43.01	23.59	22.68	51.58	37.13	–	40.28	20.14	–
Multi-head Arch.	Sequential ➔ Parallel	**9.12**	40.27	24.69	32.45	51.55	42.00	22.19	**40.37**	31.28	–
BEV Shortcut	Yes ➔ No	6.53	38.12	22.33	32.05	**52.62**	**42.34**	**23.00**	**40.39**	**31.70**	–
Training Protocol	S+J ➔ J [303]	7.42	42.73	**25.08**	31.94	49.88	40.91	22.91	39.66	31.29	–
	S+J ➔ D+J [281]	6.07	43.43	24.75	29.24	**52.96**	41.10	20.71	35.68	28.20	–
I2M+SeaBird	–	**8.71**	43.19	**25.95**	**35.76**	52.22	**43.99**	**23.23**	39.61	**31.42**	–

Mono3D performance. Conversely, removing detection head also diminishes the BEV segmentation performance for the segmentation model. This confirms the mututal benefit of sequential BEV segmentation on foreground objects and Mono3D.

Semantic Category in BEV Segmentation. We next analyze whether background categories play any role in Mono3D. Tab. 4.5 shows that changing the foreground (For.) categories to foreground + background (All) does not help Mono3D. This aligns with the observations of [167, 272, 303] that report lower performance on joint Mono3D and BEV segmentation with all categories. We believe this decrease happens because the network gets distracted while getting the background right. We also predict one foreground category (Car) instead of all in BEV segmentation. Tab. 4.5 shows that predicting all foreground categories in BEV segmentation is crucial for overall good Mono3D.

Multi-head Architecture. SeaBird employs a sequential architecture (Arch.) of segmentation and detection heads instead of parallel architecture. Tab. 4.5 shows that the sequential architecture outperforms the parallel one. We attribute this Mono3D boost to the explicit object localization provided by segmentation in the BEV plane.

BEV Shortcut. Sec. 4.3.4 mentions that SeaBird's Mono3D head utilizes both the BEV segmentation map and BEV features. Tab. 4.5 demonstrates that providing BEV features to the detection head is crucial for good Mono3D. This is because the BEV map lacks elevation information, and incorporating BEV features helps estimate elevation.

Training Protocol. SeaBird trains segmentor first and then jointly trains detector and segmentor

Table 4.6 **nuScenes Test detection results.** SeaBird pipelines achieve the best AP_{Lrg} among methods without Class Balanced Guided Sampling (CBGS) [308] and future frames. Results are from the nuScenes leaderboard or corresponding chapters on V2-99 or R101 backbones. [Key: **Best**, **Second Best**, S= Small, *= Reimplementation, §= CBGS, ○●= Future Frames.]

Resolution	Method	BBone	Venue	AP_{Lrg} (↑)	AP_{Car} (↑)	AP_{Sml} (↑)	mAP (↑)	NDS (↑)
512×1408	BEVDepth [127] in [101]	R101	AAAI23	–	–	–	39.6	48.3
	BEVStereo [125] in [101]	R101	AAAI23	–	–	–	40.4	50.2
	P2D [101]	R101	ICCV23	–	–	–	43.6	53.0
	BEVerse-S [303]	Swin-S	ArXiv	24.4	60.4	47.0	39.3	53.1
	HoP* [311]	R101	ICCV23	**36.0**	**65.0**	**53.9**	**47.9**	**57.5**
	HoP+SeaBird	R101	CVPR24	**36.6**	**65.8**	**54.7**	**48.6**	**57.0**
640×1600	SpatialDETR [53]	V2-99	ECCV22	30.2	61.0	48.5	42.5	48.7
	3DPPE [219]	V2-99	ICCV23	–	–	–	46.0	51.4
	$X3KD_{all}$ [105]	R101	CVPR23	–	–	–	45.6	56.1
	PETRv2 [150]	V2-99	ICCV23	36.4	66.7	55.6	49.0	58.2
	VEDet [29]	V2-99	CVPR23	**37.1**	68.5	**57.7**	50.5	58.5
	FrustumFormer [256]	V2-99	CVPR23	–	–	–	**51.6**	58.9
	MV2D [259]	V2-99	ICCV23	–	–	–	**51.1**	**59.6**
	HoP* [311]	V2-99	ICCV23	**37.1**	**68.7**	55.6	49.4	58.9
	HoP+SeaBird	V2-99	CVPR24	**38.4**	**70.2**	**57.4**	**51.1**	**59.7**
	SA-BEV§ [299]	V2-99	ICCV23	40.5	68.9	60.5	53.3	62.4
	FB-BEV§ [134]	V2-99	ICCV23	39.3	71.7	61.6	53.7	62.4
	CAPE§ [273]	V2-99	CVPR23	41.3	71.4	63.3	55.3	62.8
	SparseBEV○● [142]	V2-99	ICCV23	45.6	76.3	68.8	60.3	67.5
900×1600	ParametricBEV [283]	R101	ICCV23	–	–	–	46.8	49.5
	UVTR [126]	R101	NeurIPS22	35.1	67.3	52.9	47.2	55.1
	BEVFormer [132]	V2-99	ECCV22	34.4	67.7	55.2	48.9	56.9
	PolarFormer [95]	V2-99	AAAI23	36.8	68.4	55.5	49.3	57.2
	STXD [91]	V2-99	NeurIPS23	–	–	–	49.7	58.3

Table 4.7 **nuScenes Val detection results.** SeaBird pipelines outperform the two baselines BEVerse and HoP, particularly for large objects. We train all models without CBGS. See Tab. D.9 for a detailed comparison. [Key: S= Small, T= Tiny, ♙= Released, *= Reimplementation]

Resolution	Method	BBone	Venue	AP_{Lrg} (↑)	AP_{Car} (↑)	AP_{Sml} (↑)	mAP (↑)	NDS (↑)
256×704	BEVerse-T♙ [303]	Swin-T	ArXiv	18.5	53.4	38.8	32.1	46.6
	+SeaBird		CVPR24	19.5 (**+1.0**)	54.2 (**+0.8**)	41.1 (**+2.3**)	33.8 (**+1.5**)	48.1 (**+1.7**)
	HoP♙ [311]	R50	ICCV23	27.4	57.2	46.4	39.9	50.9
	+SeaBird		CVPR24	28.2 (**+0.8**)	58.6 (**+1.4**)	47.8 (**+1.4**)	41.1 (**+1.2**)	51.5 (**+0.6**)
512×1408	BEVerse-S♙ [303]	Swin-S	ArXiv	20.9	56.2	42.2	35.2	49.5
	+SeaBird		CVPR24	24.6 (**+3.7**)	58.7 (**+2.5**)	45.0 (**+2.8**)	38.2 (**+3.0**)	51.3 (**+1.8**)
	HoP* [311]	R101	ICCV23	31.4	63.7	52.5	45.2	55.0
	+SeaBird		CVPR24	32.9 (**+1.5**)	65.0 (**+1.3**)	53.1 (**+0.6**)	46.2 (**+1.0**)	54.7 (–0.3)
640×1600	HoP* [311]	V2-99	ICCV23	36.5	69.1	56.1	49.6	58.3
	+SeaBird		CVPR24	40.3 (**+3.8**)	71.7 (**+2.6**)	58.8 (**+2.7**)	52.7 (**+3.1**)	60.2 (**+1.9**)

(S+J). We compare with direct joint training (J) of [303] and training detection followed by joint training (D+J) of [281]. Tab. 4.5 shows that SeaBird training protocol works best.

4.4.3 nuScenes Mono3D

We next benchmark SeaBird on nuScenes [22], which encompasses more diverse object categories such as trailers, buses, cars and traffic cones, compared to KITTI-360 [136].

nuScenes Test. Tab. 4.6 presents the results of incorportaing SeaBird to the HoP models with the V2-99 and R101 backbones. SeaBird with both V2-99 and R101 backbones outperform several SoTA methods on the nuScenes leaderboard, as well as the baseline HoP, on nearly every metric. Interestingly, SeaBird pipelines also outperform several baselines which use higher resolution (900×1600) inputs. Most importantly, SeaBird pipelines achieve the highest AP_{Lrg} performance, providing empirical support for the claims of Th. 2.

nuScenes Val. Tab. 4.7 showcases the results of integrating SeaBird with BEVerse [303] and HoP [311] at multiple resolutions, as described in [303,311]. Tab. 4.7 demonstrates that integrating SeaBird consistently improves these detectors on almost every metric at multiple resolutions. The improvements on AP_{Lrg} empirically support the claims of Th. 2 and validate the effectiveness of dice loss and BEV segmentation in localizing large objects.

4.5 Conclusions

This chapter highlights the understudied problem of Mono3D generalization to large objects. Our findings reveal that modern frontal detectors struggle to generalize to large objects even when trained on balanced datasets. To bridge this gap, we investigate the regression and dice losses, examining their robustness under varying error levels and object sizes. We mathematically prove that the dice loss outperforms regression losses in noise-robustness and model convergence for large objects for a simplified case. Leveraging our theoretical insights, we propose SeaBird (Segmentation in Bird's View) as the first step towards generalizing to large objects. SeaBird effectively integrates BEV segmentation with the dice loss for Mono3D. SeaBird achieves SoTA results on the KITTI-360 leaderboard and consistently improves existing detectors on the nuScenes leaderboard, particularly for large objects. We hope that this initial step towards generalization will contribute to safer AVs.

Limitation. SeaBird does not fully solve the problem of generalization to large objects.

CHAPTER 5

CHARM3R: TOWARDS CAMERA HEIGHT AGNOSTIC MONOCULAR 3D OBJECT DETECTOR

To this end, we attempt generalizing Mono3D networks to occlusion, dataset and object sizes. Monocular 3D object detectors, while effective on data from one ego camera height, struggle with unseen or out-of-distribution camera heights. Existing methods often rely on Plucker embeddings, image transformations or data augmentation. This chapter takes a step towards this understudied problem by investigating the impact of camera height variations on state-of-the-art (SoTA) Mono3D models. With a systematic analysis on the extended CARLA dataset with multiple camera heights, we observe that depth estimation is a primary factor influencing performance under height variations. We mathematically prove and also empirically observe consistent negative and positive trends in mean depth error of regressed and ground-based depth models, respectively, under camera height changes. To mitigate this, we propose Camera Height Robust Monocular 3D Detector (CHARM3R), which averages both depth estimates within the model. CHARM3R significantly improves generalization to unseen camera heights, achieving SoTA performance on the CARLA dataset.

5.1 Introduction

Monocular 3D object detection (Mono3D) task uses a single image to determine both the 3D location and dimensions of objects. This technology is essential for augmented reality [2, 172, 183, 293], robotics [213], and self-driving cars [108, 132, 181], where accurate 3D understanding of the environment is crucial. Our research specifically focuses on using 3D object detectors applied to autonomous vehicles (AVs), as they have unique challenges and requirements.

AVs necessitate detectors that are robust to a wide range of intrinsic and extrinsic factors, including intrinsics [14], domains [133], object size [110], rotations [177, 307], weather conditions [137, 179], and adversarial examples [310]. Existing research primarily focusses on generalizing object detectors to these failure modes. However, this work investigates the generalization of Mono3D to another type, which, thus far, has been relatively understudied in the literature –

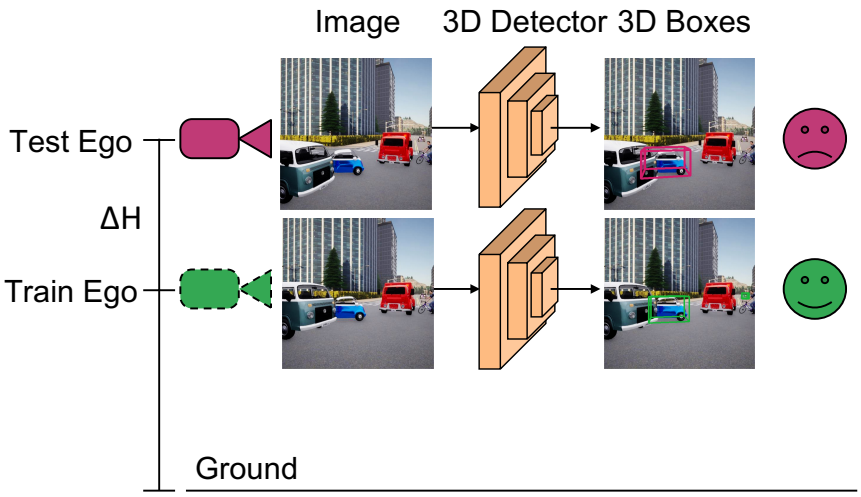
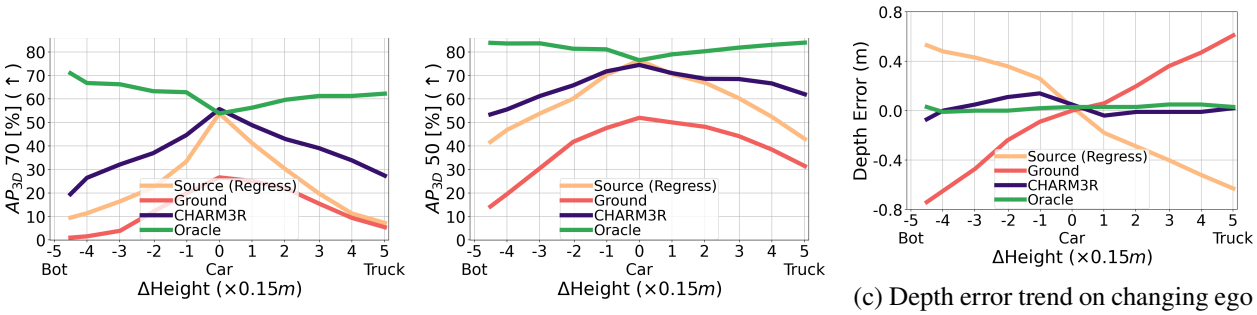


Figure 5.1 **Teaser.** Changing ego height at inference quickly **drops** Mono3D performance of SoTA detectors. A height change ΔH of $0.76m$ in inference drops AP_{3D} [%] by absolute 35 points.



(a) AP_{3D} 70 [%] Results.

(b) AP_{3D} 50 [%] Results.

(c) Depth error trend on changing ego heights.

Figure 5.2 **Performance Comparison.** The performance of SoTA detector GUP Net [159] drops significantly with changing ego heights in inference. Ground-based model shows contrasting depth error (extrapolation) trend compared to regression-based depth models. Our proposed **CHARM3R exhibits greater robustness** to such variations by averaging regression and ground-based depth estimates. All methods, except the Oracle, are trained on car-height data $\Delta H = 0m$ and tested on data from bot to truck heights.

Mono3D generalization to unseen ego camera heights.

The ego height of autonomous vehicles (AVs) varies significantly across different platforms and deployment scenarios. While almost all training data is collected from a specific ego height, such as that of a passenger car, AVs are now deployed with substantially different ego height such as small bots or trucks. Collecting, labeling datasets and retraining models for each possible height is not scalable [104], computationally expensive and impractical. Therefore, our work aims to address the challenge of *generalizing* Mono3D models to *unseen* ego heights.

Generalizing Mono3D to unseen ego heights from **single ego height** data is challenging due to

the following five reasons. First, neural models excel at In-Domain (ID) generalization, but struggle with unseen Out-Of-Domain (OOD) generalization [237, 276]. Second, ego height changes induce projective transformations [76] that CNNs [43], DEVIANT [108] or ViT [55] backbones do not effectively handle [212]. Third, existing projective equivariant backbones [168, 176] are limited to single-transform-per-image scenarios, while every pixel in a driving image undergoes a different depth-dependent transform. Fourth, the non-linear nature [21, 76] of projective transformations makes interpolation difficult. Finally, disentangled learning does not work for this problem since such approaches need at least two height data, while the training data here is from **single height**. Note that the generalization from single height to multi heights is more practical since multi-height data is unavailable in almost all real datasets.

We first systematically analyze and quantify the impact of ego height on the performance of Mono3D models trained on a single ego height. Leveraging the extended CARLA dataset [104], we evaluate the performance of state-of-the-art (SoTA) Mono3D models under multiple ego heights. Our analysis reveals that SoTA Mono3D models exhibit significant performance degradation when faced with large height changes in inference (Figs. 5.2a and 5.2b). Additionally, we empirically observe a consistent negative trend in the regressed object depth under height changes (Fig. 5.2c). Furthermore, we decompose the performance impact into individual sub-tasks and identify depth estimation as the primary contributor to this degradation.

Recent papers address ego height changes by using Plucker embeddings [4], transforming target-height images to the original height, assuming constant depth [129], or by retraining with augmented data [104]. While these techniques do offer some effectiveness, image transformation fails (Fig. 5.6) under significant height changes due to real-world depth variations. The augmentation strategy requires complicated pipelines for data synthesis at target heights and also falls short when the target height is OOD or when the target height is unknown apriori during training.

To effectively generalize Mono3D to unseen ego heights, a detector should first disentangle the depth representation from ego parameters in training and produce a new representation with new ego parameters in inference, while also canceling the trends. We propose using the projected

bottom 3D center and ground depth in addition to the regressed depth. While the ground depth is easily calculated from ego parameters and height, and can be changed based on the ego height, its direct application to Mono3D models is sub-optimal (a reason why ground plane is not used alone). However, we observe a consistent positive trend in ground depth, which contrasts with the negative trend in regressed depths. By averaging both depth estimates within the model, we effectively cancel these opposing trends and improve Mono3D generalization to unseen ego heights.

In summary the main contributions of this work include:

- We attempt the understudied problem of OOD ego height robustness in Mono3D models from single height data.
- We mathematically prove systematic negative and positive trends in the regressed and ground-based object depths, respectively, under ego height changes under simplified assumptions (Th. 3 and 4).
- We propose simple averaging of these depth estimates within the model to effectively counteract these opposing trends and generalize to unseen ego heights (Sec. 5.4.3).
- We empirically demonstrate SoTA robustness to unseen ego height changes on the CARLA dataset (Tab. 5.2).

5.2 Related Works

Extrapolation / OOD Generalization. Neural models excel at ID generalization, but struggle at OOD generalization [237, 276]. There are two major classes of methods for good OOD classification. The first does not use target data and relies on diversifying data [235], features [240, 286], predictions [119], gradients [207, 236] or losses [193, 208, 210]. Another class finetunes on small target data [103]. None of these papers attempt OOD generalization for regression tasks.

Mono3D. Mono3D has gained significant popularity, offering a cost-effective and efficient solution for perceiving the 3D world. Unlike its more expensive LiDAR and radar counterparts [155, 215, 290], or its computationally intensive stereo-based cousins [34], Mono3D relies solely on a single camera or multiple cameras with little overlaps. Earlier approaches to this task [31, 186] relied on hand-crafted features, while the recent advancements use deep models. Researchers explored

a variety of approaches to improve performance, including architectural innovations [89, 275], equivariance [29, 108], losses [15, 35], uncertainty [111, 159] and depth estimation [175, 279, 301]. A few use NMS [109, 147], corrected extrinsics [307], CAD models [25, 117, 154] or LiDAR [199] in training. Other innovations include Pseudo-LiDAR [165, 254], diffusion [196, 274], BEV feature encoding [96, 131, 303] or transformer-based [24] methods with modified positional encoding [82, 219, 233], queries [36, 93, 128, 296] or query denoising [140]. Some use pixel-wise depth [88] or object-wise depth [38, 40, 141]. Many utilize temporal fusion with short [17, 150, 248, 261] or long frame history [28, 182, 311] to boost performance. A few use distillation [100, 260], stereo [125, 261] or loss [110, 148] to improve these results further. For a comprehensive overview, we redirect readers to the surveys [163, 167]. CHARM3R selects representative Mono3D models and improves their extrapolation to unseen camera heights.

Camera Parameter Robustness. While several works aim for robust LiDAR-based detectors [27, 84, 255, 277, 282], planners [285] and map generators [197], fewer studies focus on generalizing image-based detectors. Existing image-based techniques, such as self-training [130], adversarial learning [249], perspective debiasing [158], and multi-view depth constraints [26], primarily address datasets with variations in camera intrinsics and minor height differences of $0.2m$. Some papers show robustness to other camera parameters such as intrinsics [14], and rotations [177, 307]. CHARM3R specifically tackles the challenge of generalizing to scenarios with significant camera height changes, exceeding $0.7m$.

Height-Robustness. Image-based 3D detectors such as BEVHeight [94] and MonoUNI [94] train multiple detectors at different heights, but always do ID testing. Recent works address ego height changes by either using Plucker embeddings [4, 297] for video generation/pose estimation, by transforming target-height images to the original height, assuming constant depth [129] for Mono3D, or by retraining with augmented data [104] for BEV segmentation. In contrast, we investigate the contrasting extrapolation behavior of regressed and ground-based depth estimators and average them for generalizing Mono3D to unseen camera heights.

Wide Baseline Setup. Wide baseline setups are challenging due to issues like large occlusions,

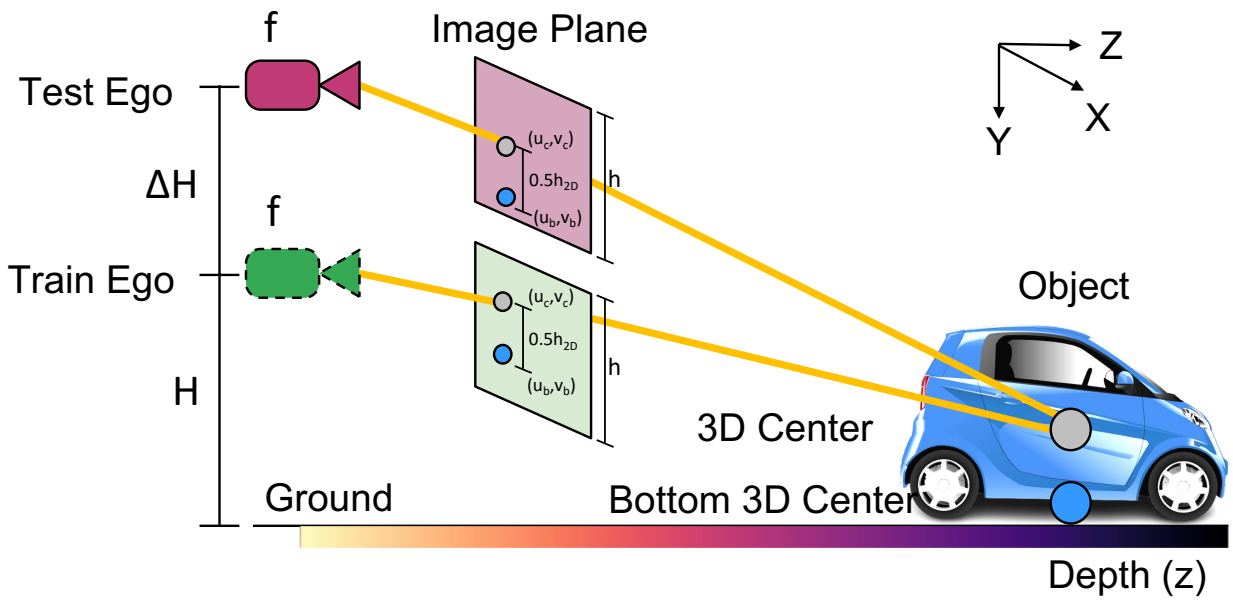


Figure 5.3 **Problem Setup**. Note that changing ego height does not change the object depth z but only its position (u_c, v_c) in the image plane. A regressed-depth model uses this pixel position to estimate the depth and therefore, fails when the ego height is changed.

depth discontinuities [229] and intensity variations [228]. Unlike traditional wide-baseline setups with arbitrary baseline movements, generalization to unseen ego height requires handling baseline movements specifically along the vertical direction.

5.3 Notations and Preliminaries

We first list out the necessary notations and preliminaries which are used throughout this chapter. These are not our contributions and can be found in the literature [65, 73, 76].

Notations. Let $\mathbf{K} \in \mathbb{R}^{3\times3}$ denote the camera intrinsic matrix, $\mathbf{R} \in \mathbb{R}^{3\times3}$ the rotation matrix and $\mathbf{T} \in \mathbb{R}^{3\times1}$ the translation vector of the extrinsic parameters. Also, $\mathbf{0} \in \mathbb{R}^{3\times1}$ denotes the zero vector in 3D. We denote the ego camera height on the car as H, and the height change relative to this car as ΔH meters. The camera intrinsics matrix $\mathbf{K}$ has focal length f and principal point (u_0, v_0). Let (u, v) represent a pixel position in the camera coordinates, and (u_c, v_c) and (u_b, v_b) denotes the projected 3D center and bottom center respectively. h denotes the height of the image plane. We show these notations pictorially in Fig. 5.3.

Pinhole Point Projection [76]. The pinhole model relates a 3D point (X, Y, Z) in the world

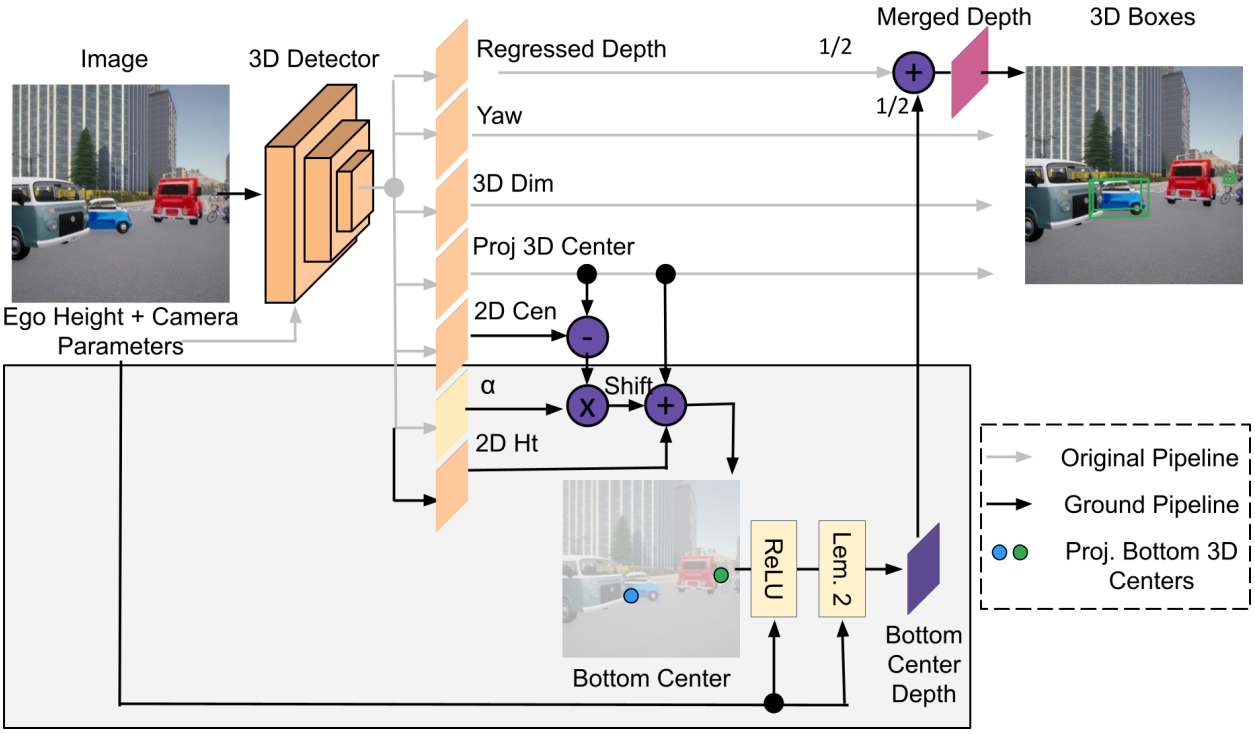


Figure 5.4 **CHARM3R Overview.** CHARM3R predicts the shift coefficient to obtain projected 3D bottom centers to query the ground depth and then averages the ground-depth and the regressed depth estimates within the model itself to output final depth estimate of a bounding box. CHARM3R uses the results of Th. 3 and 4 that demonstrate that the ground and the regressed depth models show contrasting extrapolation behaviors.

coordinate system to its 2D projected pixel (u, v) in camera coordinates as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} z = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{5.1}$$

where z denotes the depth of pixel (u, v).

Ground Depth Estimation [65,73]. While depth estimation in Mono3D is ill-posed, ground depth can be precisely determined given the camera parameters and height relative to the ground in the world coordinate system [65,73,284]. Since all datasets provide camera mounting height from the ground, we obtain the depth of ground plane pixels in closed form.

Lemma 4. ***Ground Depth of Pixel*** *[65,73,284]. Consider a pinhole camera model with intrinsics* $\mathbf{K}$*, rotation* $\mathbf{R}$ *and translation extrinsics* $\mathbf{T}$*. Let matrix* $\boldsymbol{A} = (a_{ij}) = \mathbf{R}^{-1}\mathbf{K}^{-1} \in \mathbb{R}^{3\times3}$*, and* $-\mathbf{R}^{-1}\mathbf{T}$ *as*

the vector $\boldsymbol{B} = (b_i) \in \mathbb{R}^{3\times 1}$. *Then, the ground depth* z *for a pixel* (u, v) *is*

$$z = \frac{H - b_2}{a_{21}u + a_{22}v + a_{23}}. \tag{5.2}$$

We refer to Sec. E.1.1 in the appendix for the derivation.

Lemma 5. ***Ground Depth of Pixel*** *For datasets with the rotation extrinsics* $\mathbf{R}$ *an identity, the depth estimate* z *from Lemma 4 becomes*

$$z = \frac{H - b_2}{\frac{v - v_0}{f}}. \tag{5.3}$$

We refer to Sec. E.1.2 for the proof.

5.4 CHARM3R

In this section, we first mathematically prove the contrasting extrapolation behavior of regressed and ground-based object depths under varying camera heights. To mitigate the impact of these opposing trends and improve generalization to unseen heights, we propose Camera Height Agnostic Monocular 3D Object Detector or CHARM3R. CHARM3R averages both these depth estimates within the model to mitigate these trends and improves generalization to unseen heights. Fig. 5.4 shows the overview of CHARM3R.

5.4.1 Ground-based Depth Model

Outdoor driving scenes typically contain a ground region, unlike indoor scenes. The ground depth varies with ego height, providing a valuable reference and prior for generalizing Mono3D to unseen ego heights.

Bottom Center Estimation. Lemma 4 utilizes the ground plane depth from Eq. (5.2) to estimate object depths. The numerator in Eq. (5.2) can be negative, while depth is positive for forward facing cameras. To ensure positive depth values, we apply the Rectified Linear Unit (ReLU) activation $(\max(z, 0))$ to the numerator of Eq. (5.2). This step promotes spatially continuous and meaningful ground depth representations, improving the training stability of CHARM3R. Ablation in Sec. 5.5.3 confirm the effectiveness.

In practice, CHARM3R leverages the projected 3D center (u_c, v_c), 2D height information h_{2D} and the 2D center $(u_{c,2D}, v_{c,2D})$ to compute the projected bottom 3D center (u_b, v_b) as follows:

$$u_b = u_c \;\; ; \quad v_b = v_c + \frac{1}{2} h_{2D} + \alpha(v_c - v_{c,2D}). \tag{5.4}$$

With the projected bottom center (u_b, v_b) estimated, we query the ground plane depth at this point, as derived in Lemma 4. Note that we do not use the 3D height to calculate the bottom center since projecting this point requires the box depth, which is the quantity we aim to estimate. We, now, analyze the extrapolation behavior of this ground-based depth model in the following theorem.

Theorem 3. ***Ground-based bottom center model has positive slope (trend) in extrapolation.*** *Consider a ground depth model that predicts $\hat{z}$ from the projected bottom 3D center (u_b, v_b) image. Assuming the GT object depth z is more than the ego height change ΔH, the mean depth error of the ground model exhibits a positive trend w.r.t. the height change ΔH:*

$$\mathbb{E}\left({}^{g}\hat{z}_{\Delta H} - z\right) \approx ReLU\left(\frac{1}{v_b - v_0}\right) f \Delta H, \tag{5.5}$$

where f is the focal length and (u_0, v_0) is the optical center.

Th. 3 says that the ground model over-estimates and under-estimates depth as the ego height change ΔH increases and decreases respectively.

Proof. When the ego camera shifts by ΔH m, the y-coordinate of the projected 3D bottom center v_b of a 3D box becomes $v_b + \frac{f\Delta H}{z}$. Using Eq. (5.3), the new depth ${}^{g}\hat{z}_{\Delta H}$ is

$${}^{g}\hat{z}_{\Delta H} = \frac{H + \Delta H - b_2}{\frac{v_b + \frac{f\Delta H}{z} - v_0}{f}} = \frac{H + \Delta H - b_2}{\frac{v_b - v_0}{f} + \frac{\Delta H}{z}}. \tag{5.6}$$

If the ego height change ΔH is small compared to the object depth z, $\frac{\Delta H}{z} \approx 0$. So, we write the above equation as

$${}^{g}\hat{z}_{\Delta H} \approx \frac{H + \Delta H - b_2}{\frac{v_b - v_0}{f}} = {}^{g}\hat{z}_0 + \frac{\Delta H}{\frac{v_b - v_0}{f}}$$

$$\approx z + \eta + \frac{f\Delta H}{v_b - v_0}$$
$$\implies {}^{g}\hat{z}_{\Delta H} - z \approx \eta + \frac{f\Delta H}{v_b - v_0},$$

assuming the ground depth ${}^{g}\hat{z}_0$ at train height $\Delta H = 0$ is the GT depth z added by a normal random variable η with mean 0 and variance σ^2 as in [110]. Taking expectation on both sides, the mean depth error is

$$\mathbb{E}\left({}^{g}\hat{z}_{\Delta H} - z\right) \approx \left(\frac{1}{v_b - v_0}\right) f\Delta H,$$

confirming the positive trend of the mean depth error of the ground model *w.r.t.* the height change ΔH.

The ground lies between the bottom part of the image plane/ image height (h) and the optical center y-coordinate v_0, and so $v_b - v_0 > 0$. However, in practice, it could get negative in early stage of training. To enforce non-negativity of this term, we pass $v_b - v_0$ through a ReLU non-linearity to enforce $v_b - v_0$ is positive. Sec. 5.5.3 confirms that ReLU remains important for good results. □

5.4.2 Regression-based Depth Model

Most Mono3D models rely on regression losses, to compare the predicted depth with the GT depth [108, 303]. We, next, derive the extrapolation behavior of such regressed depth model in the following theorem.

Theorem 4. ***Regressed model has negative slope (trend) in extrapolation***. *Consider a regressed depth model trained on data from single ego height, predicting depth $\hat{z}$ from the projected 3D center (u_c, v_c). Assuming a linear relationship between predicted depth and pixel position, the mean depth error of a regressed model exhibits a negative trend w.r.t. the height change ΔH:*

$$\mathbb{E}\left({}^{r}\hat{z}_{\Delta H} - z\right) = -\left(\frac{\beta}{z}\right) f\Delta H, \tag{5.7}$$

where β is a camera height independent positive constant.

Th. 4 says that regressed depth model under-estimates and over-estimates depth as the ego height change ΔH increases and decreases respectively.

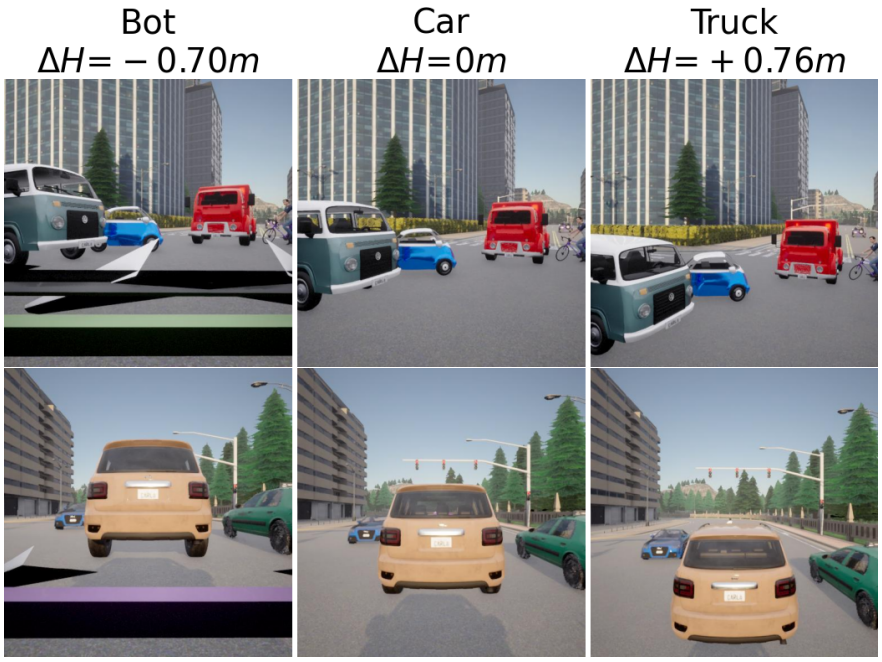


Figure 5.5 **CARLA Val samples** with both negative and positive ego height changes (ΔH) covers AVs from bots to cars to trucks.

Table 5.1 **Error analysis** of GUP Net [159] trained on $\Delta H = 0m$ on all height changes ΔH of CARLA Val split. Depth remains the biggest source of error in inference on unseen ego heights.

Oracle Params. ↓ / ΔH (m)→							AP_{3D} 70 [%] (↑)			AP_{3D} 50 [%] (↑)			MDE (m) [≈ 0]		
x	y	z	l	w	h	θ	−0.70	0	+0.76	−0.70	0	+0.76	−0.70	0	+0.76
							9.46	53.82	7.23	41.66	76.47	40.97	+0.53	+0.03	−0.63
✓							15.95	62.21	12.74	46.89	76.78	50.97	+0.53	+0.03	−0.63
	✓						13.56	59.55	10.67	44.93	76.86	49.84	+0.53	+0.03	−0.63
		✓					34.82	69.99	39.03	68.10	82.73	76.24	+0.00	+0.00	+0.00
✓	✓	✓					65.44	82.36	80.70	74.76	84.93	82.11	+0.00	+0.00	+0.00
			✓	✓	✓		10.32	56.24	7.20	42.04	76.61	42.03	+0.53	+0.03	−0.63
✓	✓	✓	✓	✓	✓		75.86	82.82	82.08	78.21	85.17	82.24	+0.00	+0.00	+0.00
✓	✓	✓	✓	✓	✓	✓	78.44	85.20	82.28	78.44	85.20	82.28	+0.00	+0.00	+0.00

Proof. Neural nets often use the y-coordinate of their projected 3D center v_c to predict depth [51]. Consider a simple linear regression model for predicting depth. Then, the regressed depth ${}^r\hat{z}_0$ is

$$
\begin{aligned}
{}^r\hat{z}_0 &= -\left(\frac{z_{max} - z_{min}}{h - v_0}\right)(v_c - v_0) + z_{max} \\
&= -\beta(v_c - v_0) + z_{max},
\end{aligned} \tag{5.8}
$$

This linear regression model has a negative slope, with a positive slope parameter β, and h being the height of the image. This model predicts depth z_{min} at pixel position $v_c = h$ and z_{max} at principal point $v_c = v_0$. When the ego camera shifts by ΔH m, the projected center of the object

becomes $v_c + \frac{f\Delta H}{z}$. Substituting this into the regression model of Eq. (5.8), we obtain the new depth ${}^r\hat{z}_{\Delta H}$ as,

$$\begin{aligned}
{}^r\hat{z}_{\Delta H} &= -\beta\left(v_c + \frac{f\Delta H}{z} - v_0\right) + z_{max} \\
&= -\beta(v_c - v_0) + z_{max} - \left(\frac{\beta}{z}\right) f\Delta H \\
&= {}^r\hat{z}_0 - \left(\frac{\beta}{z}\right) f\Delta H \\
&= z + \eta - \left(\frac{\beta}{z}\right) f\Delta H \\
\implies {}^r\hat{z}_{\Delta H} - z &= \eta - \left(\frac{\beta}{z}\right) f\Delta H,
\end{aligned}$$

assuming the regressed depth ${}^r\hat{z}_0$ at train height $\Delta H = 0$ is the GT depth z added by a normal random variable η with mean 0 and variance σ^2 as in [110]. Taking expectation on both sides, the mean depth error is

$$\mathbb{E}\left({}^r\hat{z}_{\Delta H} - z\right) = -\left(\frac{\beta}{z}\right) f\Delta H,$$

confirming the negative trend of the mean depth error of the regressed depth model *w.r.t.* the height change ΔH. □

5.4.3 Merging Depth Estimates.

Th. 3 and 4 prove that the ground and the regressed depth models show contrasting extrapolation behaviors. The former over-estimates the depth while the latter under-estimates depth as the ego height change ΔH increases. Fig. 5.4 shows how these two depth estimates are fused together. Overall, CHARM3R leverages depth information from these two source sources (with different extrapolation behaviors) to improve the Mono3D generalization to unseen camera heights. CHARM3R starts with an input image, and estimates the depth of the object using two methods: ground and regressed depth. CHARM3R outputs the projected bottom center of the object to query the ground depth (calculated from the ego camera parameters and its position and orientation relative to the ground plane as in Lemma 4). It also outputs another depth estimate based on regression. The final step combines the two estimated depths with a simple average to cancel the

Table 5.2 **CARLA Val Results.** CHARM3R **outperforms** all other baselines, especially at bigger unseen ego heights. All methods except Oracle are trained on car height $\Delta H = 0m$ and tested on bot to truck height data. [Key: **Best**]

3D Detector	Method ↓ / ΔH (m)→	AP_{3D} 70 [%] (↑)			AP_{3D} 50 [%] (↑)			MDE (m) [≈ 0]		
		−0.70	0	+0.76	−0.70	0	+0.76	−0.70	0	+0.76
GUP Net [159]	Source	9.46	53.82	7.23	41.66	76.47	40.97	+0.53	+0.03	−0.63
	Plucker [189]	8.43	55.56	10.13	37.10	**76.57**	43.22	+0.55	+0.03	−0.63
	UniDrive [129]	10.73	53.82	5.54	42.30	76.46	39.33	+0.51	+0.03	−0.67
	UniDrive++ [129]	10.83	53.82	12.27	47.81	76.46	53.08	+0.39	+0.03	−0.48
	CHARM3R	**19.45**	**55.68**	**27.33**	**53.40**	74.47	**61.98**	+0.07	+0.05	−0.02
	Oracle	70.96	53.82	62.25	83.88	76.47	83.96	+0.03	+0.03	+0.03
DEVIANT [108]	Source	8.63	50.18	6.25	40.24	73.78	41.74	+0.46	+0.01	−0.65
	Plucker [189]	8.43	**51.32**	9.52	38.24	**73.91**	44.22	+0.46	+0.01	−0.64
	UniDrive [129]	8.33	50.18	6.56	41.40	73.78	41.27	+0.46	+0.01	−0.64
	UniDrive++ [129]	6.73	50.18	12.03	42.91	73.78	52.36	+0.37	+0.01	−0.47
	CHARM3R	**17.11**	48.74	**26.24**	**49.28**	70.21	**63.60**	+0.01	+0.03	−0.02
	Oracle	71.97	50.18	62.56	84.56	73.78	83.94	+0.03	+0.01	−0.02

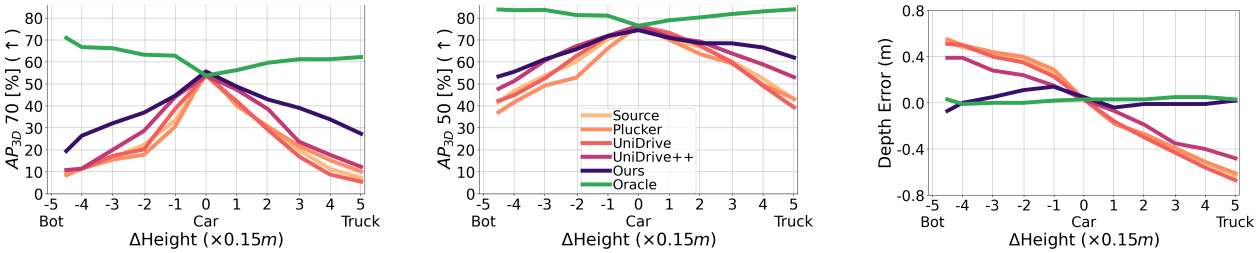
opposing trends and obtain the refined depth estimates, resulting in a set of accurate and localized 3D objects in the scene.

5.5 Experiments

Datasets. Our experiments utilize the simulated CARLA dataset[1] from [104], configured to mimic the nuScenes [22] dataset. We use this dataset for two reasons. First, this dataset reduces training and testing domain gaps, while existing public datasets lack data at multiple ego heights. Second, recent paper [104] also use this dataset for their experiments. The default CARLA dataset sweeps camera height changes ΔH from 0 to $0.76m$, rendering a dataset every $0.076m$ (car to trucks). To fully investigate the impact of camera height variations, we extend the original CARLA dataset by introducing negative height changes. The extended CARLA dataset sweeps height changes ΔH from $-0.70m$ to $0.76m$ with settings from bots to cars to trucks. Fig. 5.5 illustrates sample images from this dataset. Note that we exclude $\Delta H = -0.76m$ setting due to visibility obstructions caused by the ego vehicle's bonnet.

Data Splits. Our experiments use the *CARLA Val Split.* This dataset split [104] contains 25,000 images (2,500 scenes) from *town03* map for training and 5,000 images (500 scenes) from *town05* map for inference on multiple ego height. Except for Oracle, we train all models on training images from the car height ($\Delta H = 0m$).

[1]The authors of [104] do not release their other Nvidia-Sim dataset.



(a) $AP_{3D}\,70$ [%] comparison. (b) $AP_{3D}\,50$ [%] comparison. (c) MDE comparison.

Figure 5.6 **CARLA Val Results on GUP Net.** CHARM3R **outperforms** all baselines, especially at bigger unseen ego heights. All methods except Oracle are trained on car height and tested on all heights. Results of inference on height changes of $-0.70, 0$ and 0.76 meters are in Tab. 5.2. See Fig. 5.6 in the supplementary for another detector.

Evaluation Metrics. We choose the KITTI $AP_{3D}\,70$ percentage on the Moderate category [67] as our evaluation metric. We also report $AP_{3D}\,50$ percentage numbers following prior works [17, 109]. Additionally, we report the mean depth error (MDE) over predicted boxes with IoU_{2D} overlap greater than 0.7 with the GT boxes similar to [108]. Note that MDE is different from MAE metric of [108] that it does not take absolute value.

Detectors. We use the GUP Net [159] and DEVIANT [108] as our base detectors. The choice of these models encompasses CNN [159] and group CNN-based [108] architectures.

Baselines. We compare against the following baselines:

- *Source*: This is the Mono3D model trained on the car height ($\Delta H = 0m$) data.
- *Plucker Embeddings* [189,234]: Training a Mono3D model with Plucker embeddings to improve robustness as in 3D pose estimation and reconstruction tasks. Plucker embeddings generalize the intrinsic-focused CAM-Convs [57] embeddings to camera extrinsics.
- *UniDrive* [129]: Transforming unseen ego height (target) images to car height (source) assuming objects at fixed distance parameter ($50m$) and then passing to the Mono3D model.
- *UniDrive++* [129]: UniDrive with distance parameter optimized per dataset.
- *Oracle*: We also report the *Oracle* Mono3D model, which is trained and tested on the **same** ego height. The Oracle serves as the **upper bound** of all baselines.

Table 5.3 **CARLA Val Results with ResNet-18 backbone**. CHARM3R **outperforms** all baselines, especially at bigger unseen ego heights. All methods except Oracle are trained on car height $\Delta H = 0m$ and tested on bot to truck height data. [Key: **Best**]

3D Detector	Method ↓ / ΔH (m)→	AP_{3D} 70 [%] (↑)			AP_{3D} 50 [%] (↑)			MDE (m) [≈ 0]		
		−0.70	0	+0.76	−0.70	0	+0.76	−0.70	0	+0.76
GUP Net [159]	Source	10.13	**49.82**	5.28	47.15	**73.49**	42.70	+0.40	+0.01	−0.65
	UniDrive [129]	10.05	**49.82**	6.15	47.15	**73.49**	43.89	+0.35	+0.01	−0.62
	UniDrive++ [129]	9.37	**49.82**	13.00	52.95	**73.49**	55.57	+0.31	+0.01	−0.46
	CHARM3R	**16.62**	46.13	**24.50**	**57.00**	67.83	**60.86**	−0.15	+0.00	+0.07
	Oracle	70.25	49.82	62.93	83.49	73.49	84.07	−0.01	+0.05	+0.07
DEVIANT [108]	Source	8.83	**49.88**	4.43	42.10	72.79	38.42	+0.40	+0.01	−0.69
	UniDrive [129]	8.21	49.87	3.75	42.21	72.79	38.38	+0.40	+0.01	−0.70
	UniDrive++ [129]	6.01	49.87	12.03	43.99	72.79	50.67	+0.38	+0.01	−0.50
	CHARM3R	**14.96**	49.13	**23.66**	**52.68**	**72.95**	**60.98**	−0.07	+0.05	+0.02
	Oracle	68.35	49.88	58.49	84.03	72.79	83.42	−0.04	+0.01	−0.1

Table 5.4 **Ablation Studies** of GUP Net + CHARM3R on the CARLA Val split on unseen ego heights. [Key: **Best**]

Change	From →To / ΔH (m)→	AP_{3D} 70 [%] (↑)			AP_{3D} 50 [%] (↑)			MDE (m) [≈ 0]		
		−0.70	0	+0.76	−0.70	0	+0.76	−0.70	0	+0.76
GUP Net [159]	–	9.46	53.82	7.23	41.66	76.47	40.97	+0.53	+0.05	−0.63
Merge	Regress+Ground → Regress	9.46	53.82	7.23	41.66	76.47	40.97	+0.53	+0.05	−0.63
	Regress+Ground → Ground	0.98	26.61	5.39	14.21	51.97	31.42	−0.80	−0.01	+0.55
	Within Model → Offline	12.86	47.66	18.36	49.86	76.30	54.38	+0.24	+0.02	−0.28
	Simple Avg → Learned Avg	8.25	**56.49**	9.53	38.58	**76.82**	43.13	+0.56	−0.03	−0.62
Ground	ReLU → No ReLU	0.60	52.94	0.07	15.66	74.79	4.50	−1.09	−0.01	+1.34
Formulation	Product → Sum	3.28	37.22	12.79	17.28	63.88	47.09	+0.56	+0.09	−0.22
CHARM3R	–	**19.45**	55.68	**27.33**	**53.40**	74.47	**61.98**	−0.07	+0.05	+0.02
Oracle	–	70.96	53.82	62.25	83.88	76.47	83.96	+0.03	+0.03	+0.03

5.5.1 CARLA Error Analysis

We first report the error analysis of the baseline GUP Net [159] in Tab. 5.1 by replacing the predicted box data with the oracle parameters of the box as in [110, 166]. We consider the GT box to be an oracle box for predicted box if the euclidean distance is less than $4m$ [110]. In case of multiple GT being matched to one box, we consider the oracle with the minimum distance. Tab. 5.1 shows that depth is the biggest source of error for Mono3D task under ego height changes as also observed for single height settings in [108, 110, 166]. Note that the Oracle does not get 100% results since we only replace box parameters in the baseline and consequently, the missed boxes in the baseline are not added.

5.5.2 CARLA Height Robustness Results

Tab. 5.2 presents the CARLA Val results, reporting the **median** model over three different seeds with the model being the final checkpoint as [108]. It compares baselines and our CHARM3R on all

Mono3D models - GUP Net [159], and DEVIANT [108]. Except for Oracle, all models are trained from car height data and tested on all ego heights. Tab. 5.2 confirms that CHARM3R outperforms other baselines on all the Mono3D models, and results in a better height robust detector. We also plot these AP_{3D} numbers and depth errors visually in Fig. 5.6 for intermediate height changes to confirm our observations. The MDE comparison in Fig. 5.6c also shows the trend of baselines, while CHARM3R cancels the opposite trends in extrapolation.

Oracle Biases. We further note biases in the Oracle models at big changes in ego height. This agrees the observations of [104] in the BEV segmentation task. While higher AP_{3D} for a higher height could be explained by fewer occlusions due a higher height, higher AP_{3D} at lower camera height is not explained by this hypothesis. We leave the analysis of higher Oracle numbers for a future work.

Results on Other Backbone. We next investigate whether the extrapolation behavior holds for other backbones as well following DEVIANT [108]. So, we benchmark on the ResNet-18 backbone. Tab. 5.3 results show that extrapolation shows up in other backbones and CHARM3R again outperforms all baselines. The biggest gains are in big camera height changes, which is consistent with Tab. 5.2 results.

5.5.3 Ablation Studies

Tab. 5.4 ablates the design choices of GUP Net + CHARM3R on CARLA Val split, with the experimental setup of Sec. 5.5.

Depth Merge. We first analyze the impact of averaging the two depth estimates. Merging both regressed and ground-based depth estimates is crucial for optimal performance. Relying solely on the regressed depth gives good ID performance but bad OOD performance. Using only ground depth generalizes poorly in both ID and OOD settings, which is why it is not used in modern Mono3D models. However, it has a contrasting extrapolation MDE compared to regression models. While offline merging of depth estimates from regression-only and ground-only models also improves extrapolation, it is slower and lacks end-to-end training. We also experiment with changing the simple averaging of CHARM3R to learned averaging. Simple average of CHARM3R outperforms

learned one in OOD test because the learned average overfits to train distribution.

ReLUed Ground. Sec. 5.4.1 says that ReLU activation applied to the ground depth ensures spatial continuity and improves model training stability. Removing the ReLU leads to training instability and suboptimal extrapolation to camera height. (The training also collapses in some cases).

Formulation. CHARM3R estimates the projected 3D bottom center by using the projected 3D center and the 2D height prediction. Eq. (5.4) predicts a coefficient α to determine the precise bottom center location. Product means predicting α and then multiplying by $(v_c - v_{c,2D})$ to obtain the shift, while sum means directly predicting the shift α. Replace this product formulation by the sum formulation of α confirms that the product is more effective than the sum.

5.6 Conclusions

This chapter highlights the understudied problem of Mono3D generalization to unseen ego heights. We first systematically analyze the impact of camera height variations on state-of-the-art Mono3D models, identifying depth estimation as the primary factor affecting performance. We mathematically prove and also empirically observe consistent negative and positive trends in regressed and ground-based object depth estimates, respectively, under camera height changes. This chapter then takes a step towards generalization to unseen camera heights and proposes CHARM3R. CHARM3R averages both depth estimates within the model to mitigate these opposing trends. CHARM3R significantly enhances the generalization of Mono3D models to unseen camera heights, achieving SoTA performance on the CARLA dataset. We hope that this initial step towards generalization will contribute to safer AVs. Future work involves extending this idea to more Mono3D models.

Limitation. CHARM3R does not fully solve the generalization issue to unseen camera heights.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we attempt generalizing Mono3D networks to occlusion, dataset, object sizes and camera heights. The backbones of our models is in all cases a convolutional neural network or a transformer backbone. While the current Mono3D networks generalize fairly well across these shifts, they still suffer from the following issues:

- They do not generalize to unseen datasets during training.
- They do not multiple handle tasks like depth prediction, semantic scene completion and Mono3D.
- They do not generalize to unknown or noisy camera extrinsics.
- They do not handle multiple camera models.

Generalizing to Unseen Datasets. Current multi-dataset trained baselines such as Cube R-CNN [14] generalize poorly to datasets unseen in training. In other words, these models do not generalize in cross-dataset settings. Generalizing Mono3D to unseen datasets remains unsolved till date. We conjecture that the cause of limited generalization is the limited training data and specialized backbones which handle projective geometry.

Generalizing to Multiple Tasks. Tasks like metric depth prediction, semantic scene completion and Mono3D all represent 3D scene understanding at varying levels of granularity from points to voxels to objects. While there are networks that specialize in doing each task, a single model understands all these granularities as well as intermediate granularities remains an exciting direction for solidyfying 3D understanding and task generalization.

Generalizing to Unknown Extrinsics. Current Mono3D methods work well when trained and tested on the same extrinsics. However, such methods do not work well when the camera extrinsics, is unknown during testing. Joint Mono3D and camera calibration remains an open problem.

Generalizing to Camera Models. Current methods handle only pinhole cameras, while the cameras available today also include fisheye and 360 camera models. Generalizing Mono3D networks to handle any camera model remains another open problem in this area.

Advances in Mono3D task enable diverse applications such as Autonomous Driving, Metaverse

and robotics. The goal of home robots is to assist humans in indoor activities, such as cooking or cleaning. Future works which generalize Mono3D along these directions will make our limited 3D scene understanding even more powerful.

BIBLIOGRAPHY

[1] The KITTI Vision Benchmark Suite. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d. Accessed: 2022-07-03. 18, 35

[2] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 2018. 1, 6, 25, 44, 61

[3] Samaneh Azadi, Jiashi Feng, and Trevor Darrell. Learning detection with diverse proposals. In *CVPR*, 2017. 9, 16, 17

[4] Sherwin Bahmani, Ivan Skorokhodov, Aliaksandr Siarohin, Willi Menapace, Guocheng Qian, Michael Vasilkovsky, Hsin-Ying Lee, Chaoyang Wang, Jiaxu Zou, Andrea Tagliasacchi, David Lindell, and Sergey Tulyakov. VD3D: Taming large video diffusion transformers for 3D camera control. *arXiv preprint arXiv:2407.12781*, 2024. 63, 65

[5] Wentao Bao, Bin Xu, and Zhenzhong Chen. MonoFENet: Monocular 3D object detection with feature enhancement networks. *IEEE Transactions on Image Processing*, 2019. 9

[6] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *ICCV*, 2019. 54

[7] Deniz Beker, Hiroharu Kato, Mihai Adrian Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Monocular differentiable rendering for self-supervised 3D object detection. In *ECCV*, 2020. 20

[8] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. In *NeurIPS*, 2020. 11

[9] Zygmunt Birnbaum. An inequality for Mill's ratio. *The Annals of Mathematical Statistics*, 1942. 144

[10] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *ICML*, 2020. 11, 12

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 26

[12] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry Davis. Soft-NMS–improving object detection with one line of code. In *ICCV*, 2017. 8, 9, 10, 11, 15, 16, 17, 21, 23, 108, 109, 110

[13] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry Davis. Soft-NMS implementation. https://github.com/bharatsingh430/soft-nms/blob/master/lib/nms/cpu_nms.pyx#L98, 2017. Accessed: 2021-01-18. 10

[14] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In *CVPR*, 2023. 1, 44, 55, 56, 57, 61, 65, 78

[15] Garrick Brazil and Xiaoming Liu. M3D-RPN: Monocular 3D region proposal network for object detection. In *ICCV*, 2019. 6, 9, 16, 18, 19, 20, 21, 22, 23, 25, 26, 29, 37, 42, 48, 65, 108, 110, 121, 128, 129

[16] Garrick Brazil and Xiaoming Liu. Pedestrian detection with autoregressive network phases. In *CVPR*, 2019. 9

[17] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3D object detection in monocular video. In *ECCV*, 2020. 6, 7, 9, 16, 18, 19, 20, 21, 22, 23, 25, 29, 35, 36, 48, 65, 74, 107, 108, 109, 110, 111, 112, 113, 117, 121, 131

[18] Garrick Brazil, Xi Yin, and Xiaoming Liu. Illuminating pedestrians via simultaneous detection & segmentation. In *ICCV*, 2017. 9

[19] Michael Bronstein. Convolution from first principles. https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028. Accessed: 2021-08-13. 25, 28, 29, 114

[20] Michael Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021. 28, 29, 114

[21] Brian Burns, Richard Weiss, and Edward Riseman. The non-existence of general-case view-invariants. In *Geometric invariance in computer vision*. 1992. 29, 30, 63, 114, 115

[22] Holger Caesar, Varun Bankiti, Alex Lang, Sourabh Vora, Venice Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 34, 35, 37, 46, 53, 54, 60, 73, 128, 149, 164

[23] Brittany Caldwell. 2 die when tesla crashes into parked tractor-trailer in florida. https://www.wftv.com/news/local/2-die-when-tesla-crashes-into-parked-tractor-trailer-florida/KJGMHHYTQZA2HNAHWL2OFSVIPM/, 2022. Accessed: 2023-11-06. 2, 45

[24] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 48, 65

[25] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In *CVPR*, 2017. 29, 48, 65

[26] Gyusam Chang, Jiwon Lee, Donghyun Kim, Jinkyu Kim, Dongwook Lee, Daehyun Ji, Sujin Jang, and Sangpil Kim. Unified domain generalization and adaptation for multi-view 3D

object detection. In *NeurIPS*, 2024. 65

[27] Gyusam Chang, Wonseok Roh, Sujin Jang, Dongwook Lee, Daehyun Ji, Gyeongrok Oh, Jinsun Park, Jinkyu Kim, and Sangpil Kim. CMDA: Cross-modal and domain adversarial adaptation for LiDAR-based 3D object detection. In *AAAI*, 2024. 65

[28] Ming Chang, Xishan Zhang, Rui Zhang, Zhipeng Zhao, Guanhua He, and Shaoli Liu. RecurrentBEV: A long-term temporal fusion framework for multi-view 3D detection. In *ECCV*, 2024. 65

[29] Dian Chen, Jie Li, Vitor Guizilini, Rares Andrei Ambrus, and Adrien Gaidon. Viewpoint equivariance for multi-view 3D object detection. In *CVPR*, 2023. 48, 59, 65

[30] Kean Chen, Weiyao Lin, Jianguo Li, John See, Ji Wang, and Junni Zou. AP-Loss for accurate one-stage object detection. *TPAMI*, 2020. 17, 19

[31] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D object detection for autonomous driving. In *CVPR*, 2016. 9, 29, 47, 64

[32] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D object proposals for accurate object class detection. In *NeurIPS*, 2015. 18, 35

[33] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *CVPR*, 2017. 6, 9

[34] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. DSGN: Deep stereo geometry network for 3D object detection. In *CVPR*, 2020. 1, 47, 64

[35] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. MonoPair: Monocular 3D object detection using pairwise spatial relationships. In *CVPR*, 2020. 6, 9, 19, 20, 21, 23, 25, 29, 36, 48, 65

[36] Zhili Chen, Shuangjie Xu, Maosheng Ye, Zian Qian, Xiaoyi Zou, Dit-Yan Yeung, and Qifeng Chen. Learning high-resolution vector representation from multi-camera images for 3D object detection. In *ECCV*, 2024. 65

[37] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. NEAT: Neural attention fields for end-to-end autonomous driving. In *ICCV*, 2021. 48

[38] Wonhyeok Choi, Mingyu Shin, and Sunghoon Im. Depth-discriminative metric learning for monocular 3D object detection. In *NeurIPS*, 2023. 48, 65

[39] Zhiyu Chong, Xinzhu Ma, Hong Zhang, Yuxin Yue, Haojie Li, Zhihui Wang, and Wanli Ouyang. MonoDistill: Learning spatial features for monocular 3D object detection. In *ICLR*, 2022. 35, 36, 132

[40] Xiaomeng Chu, Jiajun Deng, Yuan Zhao, Jianmin Ji, Yu Zhang, Houqiang Li, and Yanyong

Zhang. OA-BEV: Bringing object awareness to bird's-eye-view representation for multi-camera 3D object detection. *arXiv preprint arXiv:2301.05711*, 2023. 48, 65

[41] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *ICLR*, 2018. 26, 28

[42] Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups. In *ICML*, 2014. 28

[43] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, 2016. 28, 63, 114

[44] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 149

[45] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020. 46

[46] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranks and sorting using optimal transport. In *NeurIPS*, 2019. 11

[47] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 9

[48] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 124

[49] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 2011. 9, 16, 17

[50] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *ICML*, 2016. 28, 114

[51] Tom van Dijk and Guido de Croon. How do neural networks see depth in single images? In *ICCV*, 2019. 71, 161

[52] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3D object detection. In *CVPR Workshops*, 2020. 9, 19, 20, 29, 38, 39, 121

[53] Simon Doll, Richard Schulz, Lukas Schneider, Viviane Benzin, Markus Enzweiler, and Hendrik Lensch. SpatialDETR: Robust scalable transformer-based 3D object detection from multi-view camera images with global cross-sensor attention. In *ECCV*, 2022. 59

[54] Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. Benchmarking robustness of 3D object detection to common corruptions. In *CVPR*, 2023. 1, 44

[55] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 28, 63

[56] Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. In *ICLR*, 2018. 28

[57] Jose Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. CAM-Convs: Camera-aware multi-scale convolutions for single-view depth. In *CVPR*, 2019. 74

[58] Lue Fan, Feng Wang, Naiyan Wang, and Zhao Zhang. Fully sparse 3D object detection. In *NeurIPS*, 2022. 48

[59] Chengjian Feng, Zequn Jie, Yujie Zhong, Xiangxiang Chu, and Lin Ma. AEDet: Azimuth-invariant multi-view 3D object detection. *arXiv preprint arXiv:2211.12501*, 2022. 48

[60] Roshan Fernandez. A tesla driver was killed after smashing into a firetruck on a california highway. https://www.npr.org/2023/02/20/1158367204/tesla-driver-killed-california-firetruck-nhtsa, 2023. Accessed: 2023-11-06. 2, 45

[61] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. In *NeurIPS*, 2012. 9, 29

[62] William Freeman and Edward Adelson. The design and use of steerable filters. *TPAMI*, 1991. 28, 32

[63] Kanchana Gandikota, Jonas Geiping, Zorah Lähner, Adam Czapliński, and Michael Moeller. Training or architecture? how to incorporate invariance in neural networks. *arXiv preprint arXiv:2106.10044*, 2021. 27, 38, 114

[64] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *NeurIPS*, 2017. 26, 28

[65] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. 3D-LaneNet: end-to-end 3D multiple lane detection. In *ICCV*, 2019. 66, 67

[66] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013. 112, 134

[67] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 18, 20, 34, 35, 53, 54, 74, 146

[68] Rohan Ghosh and Anupam Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. In *ICML Workshops*, 2019. 27, 28, 32, 122

[69] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 8, 9, 55

[70] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 8

[71] Gene Golub and Charles Loan. Matrix computations. 2013. 14

[72] Nikhil Gosala and Abhinav Valada. Bird's-eye-view panoptic segmentation using monocular frontal view images. *RAL*, 2022. 48, 54, 55, 57, 148, 150, 152

[73] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jinghao Miao, Jingao Wang, and Tae Eun Choe. Gen-lanenet: A generalized and scalable approach for 3D lane detection. In *ECCV*, 2020. 66, 67

[74] Adam Harley, Zhaoyuan Fang, Jie Li, Rares Ambrus, and Katerina Fragkiadaki. Simple-BEV: What really matters for multi-sensor BEV perception? In *CoRL*, 2022. 48

[75] Christopher Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988. 9

[76] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 27, 29, 30, 31, 63, 66, 116, 117

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 150

[78] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *ACCV*, 2016. 9, 18, 24

[79] Joao Henriques and Andrea Vedaldi. Warped convolutions: Efficient invariance to spatial transformations. In *ICML*, 2017. 28

[80] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. A convnet for non-maximum suppression. In *GCPR*, 2016. 7, 9, 16, 17, 18, 24

[81] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *CVPR*, 2017. 7, 9, 16, 17, 18, 24

[82] Jinghua Hou, Tong Wang, Xiaoqing Ye, Zhe Liu, Xiao Tan, Errui Ding, Jingdong Wang, and Xiang Bai. OPEN: Object-wise position embedding for multi-view 3D object detection. In *ECCV*, 2024. 65

[83] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. FIERY: future instance prediction in bird's-eye view from surround monocular cameras. In *ICCV*, 2021. 48, 50

[84] Hanjiang Hu, Zuxin Liu, Sharad Chitlangia, Akhil Agnihotri, and Ding Zhao. Investigating the impact of multi-LiDAR placement on object detection for autonomous driving. In *CVPR*, 2022. 65

[85] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3D object detection. In *CVPR*, 2020. 9

[86] Gao Huang, Zhuang Liu, Laurens Maaten, and Kilian Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 18

[87] Junjie Huang and Guan Huang. BEVDet4D: Exploit temporal cues in multi-camera 3D object detection. *arXiv preprint arXiv:2203.17054*, 2022. 55, 155

[88] Junjie Huang, Guan Huang, Zheng Zhu, Yun Ye, and Dalong Du. BEVDet: High-performance multi-camera 3D object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021. 48, 65, 155

[89] Kuan-Chih Huang, Tsung-Han Wu, Hung-Ting Su, and Winston Hsu. MonoDTR: Monocular 3D object detection with depth-aware transformer. In *CVPR*, 2022. 47, 65

[90] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. EPNet: Enhancing point features with image semantics for 3D object detection. In *ECCV*, 2020. 6, 7, 9, 23

[91] Sujin Jang, Dae Ung Jo, Sung Ju Hwang, Dongwook Lee, and Daehyun Ji. STXD: Structural and temporal cross-modal distillation for multi-view 3D object detection. In *NeurIPS*, 2023. 59

[92] Ylva Jansson and Tony Lindeberg. Scale-invariant scale-channel networks: Deep networks that generalise to previously unseen scales. *IJCV*, 2021. 27, 28, 32

[93] Haoxuanye Ji, Pengpeng Liang, and Erkang Cheng. Enhancing 3D object detection with 2D detection-guided query anchors. In *CVPR*, 2024. 65

[94] Jinrang Jia, Zhenjia Li, and Yifeng Shi. MonoUNI: A unified vehicle and infrastructure-side monocular 3D object detection network with sufficient depth clues. In *NeurIPS*, 2023. 1, 44, 65

[95] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-camera 3D object detection with polar transformers. In *AAAI*, 2023. 48, 59, 155

[96] Zheng Jiang, Jinqing Zhang, Yanan Zhang, Qingjie Liu, Zhenghui Hu, Baohui Wang, and Yunhong Wang. FSD-BEV: Foreground self-distillation for multi-view 3D object detection. In *ECCV*, 2024. 65

[97] Li Jing. *Physical symmetry enhanced neural networks*. PhD thesis, Massachusetts Institute of Technology, 2020. 28

[98] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. Locally scale-invariant convolutional neural networks. In *NeurIPS Workshops*, 2014. 28

[99] Kang Kim and Hee Lee. Probabilistic anchor assignment with IoU prediction for object

detection. In *ECCV*, 2020. 19, 24

[100] Sanmin Kim, Youngseok Kim, Sihwan Hwang, Hyeonjun Jeong, and Dongsuk Kum. LabelDistill: Label-guided cross-modal knowledge distillation for camera-based 3D object detection. In *ECCV*, 2024. 65

[101] Sanmin Kim, Youngseok Kim, In-Jae Lee, and Dongsuk Kum. Predict to Detect: Prediction-guided 3D object detection using sequential images. In *ICCV*, 2023. 59, 155

[102] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 108, 125, 150

[103] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *ICLR*, 2022. 64

[104] Tzofi Klinghoffer, Jonah Philion, Wenzheng Chen, Or Litany, Zan Gojcic, Jungseock Joo, Ramesh Raskar, Sanja Fidler, and Jose Alvarez. Towards viewpoint robustness in Bird's Eye View segmentation. In *ICCV*, 2023. 1, 44, 62, 63, 65, 73, 76, 161, 162

[105] Marvin Klingner, Shubhankar Borse, Varun Ravi Kumar, Behnaz Rezaei, Venkatraman Narayanan, Senthil Yogamani, and Fatih Porikli. X3KD: Knowledge distillation across modalities, tasks and stages for multi-camera 3D object detection. In *CVPR*, 2023. 48, 59

[106] Emile Krieken, Erman Acar, and Frank Harmelen. Analyzing differentiable fuzzy logic operators. *arXiv preprint arXiv:2002.06100*, 2020. 12, 106

[107] Jason Ku, Alex Pon, and Steven Waslander. Monocular 3D object detection leveraging accurate proposals and shape reconstruction. In *CVPR*, 2019. 19

[108] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. DEVIANT: Depth Equivariant Network for monocular 3D object detection. In *ECCV*, 2022. 1, 44, 48, 50, 53, 54, 55, 56, 57, 61, 63, 65, 70, 73, 74, 75, 76, 146, 147, 152, 153, 156, 162, 163, 164

[109] Abhinav Kumar, Garrick Brazil, and Xiaoming Liu. GrooMeD-NMS: Grouped mathematically differentiable NMS for monocular 3D object detection. In *CVPR*, 2021. 29, 33, 35, 36, 37, 48, 55, 56, 57, 65, 74, 126, 128, 132, 134, 151

[110] Abhinav Kumar, Yuliang Guo, Xinyu Huang, Liu Ren, and Xiaoming Liu. SeaBird: Segmentation in bird's view with dice loss improves monocular 3D detection of large objects. In *CVPR*, 2024. 1, 61, 65, 70, 72, 75, 162

[111] Abhinav Kumar, Tim Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. LUVLi face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In *CVPR*, 2020. 7, 29, 48, 65

[112] Animesh Kumar and Vinod Prabhakaran. Estimation of bandlimited signals from the signs of noisy samples. In *ICASSP*, 2013. 13, 33, 107

[113] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012. 49, 138, 141

[114] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. MSeg: A composite dataset for multi-domain semantic segmentation. In *CVPR*, 2020. 130

[115] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 28, 29

[116] Donghoon Lee, Geonho Cha, Ming-Hsuan Yang, and Songhwai Oh. Individualness and determinantal point processes for pedestrian detection. In *ECCV*, 2016. 9, 15, 17

[117] Hyo-Jun Lee, Hanul Kim, Su-Min Choi, Seong-Gyun Jeong, and Yeong Koh. BAAM: Monocular 3D pose and shape reconstruction with bi-contextual attention module and attention-guided modeling. In *CVPR*, 2023. 48, 65

[118] Jin Lee, Myung Han, Dong Ko, and Il Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 130

[119] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. Diversify and disambiguate: Learning from underspecified data. In *ICLR*, 2022. 64

[120] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *IJRR*, 2018. 6

[121] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. GS3D: An efficient 3D object detection framework for autonomous driving. In *CVPR*, 2019. 6, 19

[122] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3D object detection for autonomous driving. In *CVPR*, 2019. 9

[123] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. RTM3D: Real-time monocular 3D detection from object keypoints for autonomous driving. In *ECCV*, 2020. 6, 9, 19, 25, 29

[124] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. In *ACL*, 2019. 12, 106

[125] Yinhao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. BEVStereo: Enhancing depth estimation in multi-view 3D object detection with dynamic temporal stereo. In *AAAI*, 2023. 48, 59, 65

[126] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3D object detection. In *NeurIPS*, 2022. 59

[127] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. BEVDepth: Acquisition of reliable depth for multi-view 3D object detection.

In *AAAI*, 2023. 59, 155

[128] Yangguang Li, Bin Huang, Zeren Chen, Yufeng Cui, Feng Liang, Mingzhu Shen, Fenggang Liu, Enze Xie, Lu Sheng, Wanli Ouyang, and Jing Shao. Fast-BEV: A fast and strong bird's-eye view perception baseline. In *NeurIPS Workshops*, 2023. 48, 65

[129] Ye Li, Wenzhao Zheng, Xiaonan Huang, and Kurt Keutzer. UniDrive: Towards universal driving perception across camera configurations. *arXiv preprint arXiv:2410.13864*, 2024. 63, 65, 73, 74, 75, 163

[130] Zhenyu Li, Zehui Chen, Ang Li, Liangji Fang, Qinhong Jiang, Xianming Liu, and Junjun Jiang. Unsupervised domain adaptation for monocular 3D object detection via self-training. In *ECCV*, 2022. 65

[131] Zhenxin Li, Shiyi Lan, Jose Alvarez, and Zuxuan Wu. BEVNeXt: Reviving dense BEV frameworks for 3D object detection. In *CVPR*, 2024. 65

[132] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022. 1, 44, 46, 48, 53, 59, 61, 147, 155

[133] Zhuoling Li, Xiaogang Xu, SerNam Lim, and Hengshuang Zhao. Unimode: Unified monocular 3D object detection. In *CVPR*, 2024. 61

[134] Zhiqi Li, Zhiding Yu, Wenhai Wang, Anima Anandkumar, Tong Lu, and Jose Alvarez. FB-BEV: BEV representation from forward-backward view transformations. In *ICCV*, 2023. 59

[135] Qing Lian, Botao Ye, Ruijia Xu, Weilong Yao, and Tong Zhang. Geometry-aware data augmentation for monocular 3D object detection. *arXiv preprint arXiv:2104.05858*, 2021. 26, 29

[136] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D. *TPAMI*, 2022. 45, 46, 53, 54, 60, 148, 149

[137] Hongbin Lin, Yifan Zhang, Shuaicheng Niu, Shuguang Cui, and Zhen Li. MonoTTA: Fully test-time adaptation for monocular 3D object detection. In *ECCV*, 2024. 61

[138] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 39, 123, 149

[139] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *TPAMI*, 2018. 8, 9

[140] Feng Liu, Tengteng Huang, Qianjing Zhang, Haotian Yao, Chi Zhang, Fang Wan, Qixiang Ye, and Yanzhao Zhou. Ray Denoising: Depth-aware hard negative sampling for multi-view 3D object detection. In *ECCV*, 2024. 65

[141] Feng Liu and Xiaoming Liu. Voxel-based 3D detection and reconstruction of multiple objects from a single image. In *NeurIPS*, 2021. 48, 65

[142] Haisong Liu Liu, Yao Teng Teng, Tao Lu, Haiguang Wang, and Limin Wang. SparseBEV: High-performance sparse 3D object detection from multi-camera videos. In *ICCV*, 2023. 59, 154

[143] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3D object detection. In *CVPR*, 2019. 9, 19, 25, 29

[144] Lijie Liu, Chufan Wu, Jiwen Lu, Lingxi Xie, Jie Zhou, and Qi Tian. Reinforced axial refinement network for monocular 3D object detection. In *ECCV*, 2020. 19

[145] Songtao Liu, Di Huang, and Yunhong Wang. Adaptive NMS: Refining pedestrian detection in a crowd. In *CVPR*, 2019. 9, 16, 17

[146] Xianpeng Liu, Nan Xue, and Tianfu Wu. Learning auxiliary monocular contexts helps monocular 3D object detection. In *AAAI*, 2022. 29

[147] Xianpeng Liu, Ce Zheng, Kelvin Cheng, Nan Xue, Guo-Jun Qi, and Tianfu Wu. Monocular 3D object detection with bounding box denoising in 3D by perceiver. In *ICCV*, 2023. 48, 65

[148] Xianpeng Liu, Ce Zheng, Ming Qian, Nan Xue, Chen Chen, Zhebin Zhang, Chen Li, and Tianfu Wu. Multi-view attentive contextualization for multi-view 3D object detection. In *CVPR*, 2024. 65

[149] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETR: Position embedding transformation for multi-view 3D object detection. In *ECCV*, 2022. 155

[150] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETRv2: A unified framework for 3D perception from multi-camera images. In *ICCV*, 2023. 48, 59, 65, 155

[151] Yuxuan Liu, Yuan Yixuan, and Ming Liu. Ground-aware monocular 3D object detection for autonomous driving. *Robotics and Automation Letters*, 2021. 26, 35, 36

[152] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 150

[153] Zechen Liu, Zizhang Wu, and Roland Tóth. SMOKE: Single-stage monocular 3D object detection via keypoint estimation. In *CVPR Workshops*, 2020. 19

[154] Zongdai Liu, Dingfu Zhou, Feixiang Lu, Jin Fang, and Liangjun Zhang. AutoShape: Real-time shape-aware monocular 3D object detection. In *ICCV*, 2021. 29, 35, 48, 65

[155] Yunfei Long, Abhinav Kumar, Daniel Morris, Xiaoming Liu, Marcos Castro, and Punarjay Chakravarty. RADIANT: RADar Image Association Network for 3D object detection. In

AAAI, 2023. 1, 47, 64

[156] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 150, 151

[157] David Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 9

[158] Hao Lu, Yunpeng Zhang, Qing Lian, Dalong Du, and Yingcong Chen. Towards generalizable multi-camera 3D object detection via perspective debiasing. *arXiv preprint arXiv:2310.11346*, 2023. 65

[159] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3D object detection. In *ICCV*, 2021. 25, 26, 29, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 48, 50, 55, 56, 57, 62, 65, 71, 73, 74, 75, 76, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 146, 152, 153, 155, 163, 165, 166, 167

[160] Shujie Luo, Hang Dai, Ling Shao, and Yong Ding. M3DSSD: Monocular 3D single stage object detector. In *CVPR*, 2021. 19

[161] Zhipeng Luo, Changqing Zhou, Gongjie Zhang, and Shijian Lu. DETR4D: Direct multi-view 3D object detection with sparse attention. *arXiv preprint arXiv:2212.07849*, 2022. 48

[162] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking Pseudo-LiDAR representation. In *ECCV*, 2020. 29, 42

[163] Xinzhu Ma, Wanli Ouyang, Andrea Simonelli, and Elisa Ricci. 3D object detection from images for autonomous driving: A survey. *TPAMI*, 2023. 29, 48, 65

[164] Xinzhu Ma, Yongtao Wang, Yinmin Zhang, Zhiyi Xia, Yuan Meng, Zhihui Wang, Haojie Li, and Wanli Ouyang. Towards fair and comprehensive comparisons for image-based 3D object detection. In *ICCV*, 2023. 48

[165] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving. In *ICCV*, 2019. 19, 29, 48, 65

[166] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3D object detection. In *CVPR*, 2021. 26, 34, 36, 53, 55, 56, 57, 75, 119, 121, 122, 126, 152

[167] Yuexin Ma, Tai Wang, Xuyang Bai, Huitong Yang, Yuenan Hou, Yaming Wang, Yu Qiao, Ruigang Yang, Dinesh Manocha, and Xinge Zhu. Vision-centric BEV perception: A survey. *arXiv preprint arXiv:2208.02797*, 2022. 46, 48, 58, 65

[168] Lachlan MacDonald, Sameera Ramasinghe, and Simon Lucey. Enabling equivariance for arbitrary lie groups. In *CVPR*, 2022. 63

[169] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10D: Monocular lifting of 2D detection to 6D pose and metric shape. In *CVPR*, 2019. 19

[170] Diego Marcos, Benjamin Kellenberger, Sylvain Lobry, and Devis Tuia. Scale equivariance in CNNs with vector fields. In *ICML Workshops*, 2018. 28

[171] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *ICCV*, 2017. 28

[172] Nathaniel Merrill, Yuliang Guo, Xingxing Zuo, Xinyu Huang, Stefan Leutenegger, Xi Peng, Liu Ren, and Guoquan Huang. Symmetry and uncertainty-aware object SLAM for 6DoF object pose estimation. In *CVPR*, 2022. 1, 44, 61

[173] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 2009. 28

[174] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *IJCV*, 2004. 9

[175] Zhixiang Min, Bingbing Zhuang, Samuel Schulter, Buyu Liu, Enrique Dunn, and Manmohan Chandraker. NeurOCS: Neural NOCS supervision for monocular 3D object localization. In *CVPR*, 2023. 48, 65

[176] Mircea Mironenco and Patrick Forré. Lie group decompositions for equivariant neural networks. In *ICLR*, 2024. 63

[177] SungHo Moon, JinWoo Bae, and SungHoon Im. Rotation matters: Generalized monocular 3D object detection for various camera systems. *arXiv preprint arXiv:2310.05366*, 2023. 1, 44, 61, 65

[178] Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3D LiDAR data in non-flat urban environments using a local convexity criterion. In *Intelligent Vehicles Symposium*, 2009. 9

[179] Youngmin Oh, Hyung-Il Kim, Seong Tae Kim, and Jung Kim. MonoWAD: Weather-adaptive diffusion model for robust monocular 3D object detection. In *ECCV*, 2024. 61

[180] Bowen Pan, Jiankai Sun, Ho Leung, Alex Andonian, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. *RAL*, 2020. 48

[181] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is Pseudo-LiDAR needed for monocular 3D object detection? In *ICCV*, 2021. 1, 29, 35, 36, 37, 44, 61, 127, 132, 150

[182] Jinhyung Park, Chenfeng Xu, Shijia Yang, Kurt Keutzer, Kris Kitani, Masayoshi Tomizuka, and Wei Zhan. Time will tell: New outlooks and a baseline for temporal multi-view 3D object detection. In *ICLR*, 2023. 48, 65, 155

[183] Kiru Park, Timothy Patten, and Markus Vincze. Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In *ICCV*, 2019. 1, 44, 61

[184] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 18, 34, 149

[185] Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris Maddison. Gradient estimation with stochastic softmax tricks. In *NeurIPS*, 2020. 11, 15

[186] Nadia Payet and Sinisa Todorovic. From contours to 3D object detection and pose estimation. In *ICCV*, 2011. 9, 29, 47, 64

[187] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Multi-view and 3D deformable part models. *TPAMI*, 2015. 9, 29

[188] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D. In *ECCV*, 2020. 48

[189] Julius Plücker. *Analytisch-geometrische Entwicklungen*. GD Baedeker, 1828. 73, 74

[190] Marin Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *ICLR*, 2019. 11

[191] Sebastian Prillo and Julian Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *ICML*, 2020. 11, 12

[192] Sergey Prokudin, Daniel Kappler, Sebastian Nowozin, and Peter Gehler. Learning to filter object detections. In *GCPR*, 2017. 6, 7, 8, 9, 11, 12, 16, 17, 18, 24, 105

[193] Aahlad Manas Puli, Lily Zhang, Yoav Wald, and Rajesh Ranganath. Don't blame dataset shift! shortcut learning due to gradients and cross entropy. In *NeurIPS*, 2023. 64

[194] Charles Qi, Or Litany, Kaiming He, and Leonidas Guibas. Deep hough voting for 3D object detection in point clouds. In *ICCV*, 2019. 56

[195] Zengyi Qin, Jinglu Wang, and Yan Lu. MonoGRNet: A geometric reasoning network for 3D object localization. In *AAAI*, 2019. 19, 20

[196] Yasiru Ranasinghe, Deepti Hegde, and Vishal M Patel. MonoDiff: Monocular 3D object detection and pose estimation with diffusion models. In *CVPR*, 2024. 65

[197] Narayanan Elavathur Ranganatha, Hengyuan Zhang, Shashank Venkatramani, Jing-Yan Liao, and Henrik Christensen. SemVecNet: Generalizable vector map generation for arbitrary sensor configurations. 2024. 65

[198] Matthias Rath and Alexandru Condurache. Boosting deep neural networks with geometrical prior knowledge: A survey. *arXiv preprint arXiv:2006.16867*, 2020. 25, 28, 29, 114

[199] Cody Reading, Ali Harakeh, Julia Chae, and Steven Waslander. Categorical depth distribution network for monocular 3D object detection. In *CVPR*, 2021. 29, 35, 36, 42, 48, 65, 124, 126, 132

[200] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 8, 9

[201] Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steve Seitz. Soccer on your tabletop. In *CVPR*, 2018. 6, 25

[202] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 6, 8, 9, 25

[203] David Rey, Gérard Subsol, Hervé Delingette, and Nicholas Ayache. Automatic detection and segmentation of evolving processes in 3D medical images: Application to multiple sclerosis. *Medical Image Analysis*, 2002. 6

[204] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 16, 17

[205] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *CVPR*, 2020. 48

[206] Azriel Rosenfeld and Mark Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, 1971. 9

[207] Andrew Slavin Ross, Weiwei Pan, and Finale Doshi-Velez. Learning qualitatively diverse and interpretable rules for classification. In *ICML Workshops*, 2018. 64

[208] Shouwei Ruan, Yinpeng Dong, Hang Su, Jianteng Peng, Ning Chen, and Xingxing Wei. Towards viewpoint-invariant visual recognition via adversarial training. In *ICCV*, 2023. 64

[209] Sitapa Rujikietgumjorn and Robert Collins. Optimized pedestrian detection for multiple and occluded people. In *CVPR*, 2013. 9, 15, 17

[210] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *ICLR*, 2019. 64

[211] Avishkar Saha, Oscar Mendez, Chris Russell, and Richard Bowden. Translating images into maps. In *ICRA*, 2022. 48, 50, 54, 55, 56, 57, 149, 150, 152

[212] Ayush Sarkar, Hanlin Mai, Amitabh Mahapatra, Svetlana Lazebnik, David Forsyth, and Anand Bhattad. Shadows don't lie and lines can't bend! generative models don't know

projective geometry... for now. In *CVPR*, 2024. 63

[213] Ashutosh Saxena, Justin Driemeyer, and Andrew Ng. Robotic grasping of novel objects using vision. *IJRR*, 2008. 1, 6, 25, 44, 61

[214] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML*, 2007. 49, 50, 51, 138, 141

[215] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 9, 29, 47, 48, 64

[216] Xuepeng Shi, Zhixiang Chen, and Tae-Kyun Kim. Distance-normalized unified representation for monocular 3D object detection. In *ECCV*, 2020. 6, 7, 9, 15, 17, 19, 21, 23, 24, 48, 108, 109

[217] Xuepeng Shi, Zhixiang Chen, and Tae-Kyun Kim. Multivariate probabilistic monocular 3D object detection. In *WACV*, 2023. 47

[218] Xuepeng Shi, Qi Ye, Xiaozhi Chen, Chuangrong Chen, Zhixiang Chen, and Tae-Kyun Kim. Geometry-based distance decomposition for monocular 3D object detection. In *ICCV*, 2021. 26, 35, 36, 37, 128, 129, 132

[219] Changyong Shu, Fisher Yu, and Yifan Liu. 3DPPE: 3D point positional encoding for multi-camera 3D object detection transformers. In *ICCV*, 2023. 48, 59, 65, 155

[220] Andrea Simonelli, Samuel Bulò, Lorenzo Porzi, Manuel Antequera, and Peter Kontschieder. Disentangling monocular 3D object detection: From single to multi-class recognition. *TPAMI*, 2020. 6, 7, 9, 18, 19, 20, 25, 34, 35, 36, 128, 132

[221] Andrea Simonelli, Samuel Bulò, Lorenzo Porzi, Peter Kontschieder, and Elisa Ricci. Are we missing confidence in Pseudo-LiDAR methods for monocular 3D object detection? In *ICCV*, 2021. 29, 35, 36, 109, 130

[222] Andrea Simonelli, Samuel Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3D object detection. In *ICCV*, 2019. 7, 19, 20, 34, 128

[223] Andrea Simonelli, Samuel Bulò, Lorenzo Porzi, Elisa Ricci, and Peter Kontschieder. Towards generalization across depth for monocular 3D object detection. In *ECCV*, 2020. 9, 19, 20, 25, 26, 29

[224] Samik Some, Mithun Das Gupta, and Vinay Namboodiri. Determinantal point process as an alternative to NMS. In *BMVC*, 2020. 9, 15, 17

[225] Ivan Sosnovik, Artem Moskalev, and Arnold Smeulders. DISCO: accurate discrete scale convolutions. In *BMVC*, 2021. 39, 41

[226] Ivan Sosnovik, Artem Moskalev, and Arnold Smeulders. Scale equivariance improves

siamese tracking. In *WACV*, 2021. 28, 32, 34, 40, 122, 123, 124

[227] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *ICLR*, 2020. 27, 28, 31, 32, 33, 38, 39, 41, 119, 120, 122, 123, 132

[228] Christoph Strecha, Rik Fransens, and Luc Van Gool. Wide-baseline stereo from multiple views: a probabilistic account. In *CVPR*, 2004. 66

[229] Christoph Strecha, Tinne Tuytelaars, and Luc Van Gool. Dense matching of multiple wide-baseline views. In *ICCV*, 2003. 66

[230] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 34, 42, 54

[231] Mingxing Tan, Ruoming Pang, and Quoc Le. EfficientDet: Scalable and efficient object detection. In *CVPR*, 2020. 150

[232] Yunlei Tang, Sebastian Dorn, and Chiragkumar Savani. Center3D: Center-based monocular 3D object detection with joint depth understanding. *arXiv preprint arXiv:2005.13423*, 2020. 8, 9, 25, 26, 29

[233] Yingqi Tang, Zhaotie Meng, Guoliang Chen, and Erkang Cheng. SimPB: A single model for 2D and 3D object detection from multiple cameras. In *ECCV*, 2024. 65

[234] Seth Teller and Michael Hohmeyer. Determining the lines through four lines. *Journal of graphics tools*, 1999. 74

[235] Damien Teney, Ehsan Abbasnejad, and Anton Hengel. Unshuffling data for improved generalization in visual question answering. In *ICCV*, 2021. 64

[236] Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior OOD generalization. In *CVPR*, 2022. 64

[237] Damien Teney, Yong Lin, Seong Joon Oh, and Ehsan Abbasnejad. ID and OOD performance are sometimes inversely correlated on real-world datasets. In *NeurIPS*, 2023. 63, 64

[238] Sugirtha Thayalan-Vaz, Sridevi M, Khailash Santhakumar, B Ravi Kiran, Thomas Gauthier, and Senthil Yogamani. Exploring 2D data augmentation for 3D monocular object detection. *arXiv preprint arXiv:2104.10786*, 2021. 26, 29

[239] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds. *arXiv preprint arXiv:1802.08219*, 2018. 28

[240] Rishabh Tiwari and Pradeep Shenoy. Overcoming simplicity bias in deep networks using a feature sieve. In *ICML*, 2023. 64

[241] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness NMS and bounded IoU loss. In *CVPR*, 2018. 19, 24

[242] Alexandru Vasile and Richard Marino. Pose-independent automatic target detection and recognition using 3D laser radar imagery. *Lincoln laboratory journal*, 2005. 9

[243] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 9

[244] Li Wan, David Eigen, and Rob Fergus. End-to-end integration of a convolution network, deformable parts model and non-maximum suppression. In *CVPR*, 2015. 9, 16, 17

[245] Li Wang, Liang Du, Xiaoqing Ye, Yanwei Fu, Guodong Guo, Xiangyang Xue, Jianfeng Feng, and Li Zhang. Depth-conditioned dynamic message propagation for monocular 3D object detection. In *CVPR*, 2021. 36

[246] Li Wang, Li Zhang, Yi Zhu, Zhi Zhang, Tong He, Mu Li, and Xiangyang Xue. Progressive coordinate transforms for monocular 3D object detection. In *NeurIPS*, 2021. 35, 36, 42, 132

[247] Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *ICLR*, 2021. 28

[248] Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. StreamPETR: Exploring object-centric temporal modeling for efficient multi-view 3D object detection. In *ICCV*, 2023. 48, 65

[249] Shuo Wang, Xinhai Zhao, Hai-Ming Xu, Zehui Chen, Dameng Yu, Jiahao Chang, Zhen Yang, and Feng Zhao. Towards domain generalization for multi-view 3D object detection in bird-eye-view. In *CVPR*, 2023. 65

[250] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. FCOS3D: Fully convolutional one-stage monocular 3D object detection. In *ICCV Workshops*, 2021. 155

[251] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *CoRL*, 2021. 155

[252] Xueqing Wang, Diankun Zhang, Haoyu Niu, and Xiaojun Liu. Segmentation can aid detection: Segmentation-guided single stage detection for 3D point cloud. *Electronics*, 2023. 48

[253] Xinjiang Wang, Shilong Zhang, Zhuoran Yu, Litong Feng, and Wayne Zhang. Scale-equalizing pyramid convolution for object detection. In *CVPR*, 2020. 127

[254] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Weinberger. Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object

detection for autonomous driving. In *CVPR*, 2019. 9, 29, 43, 48, 65

[255] Yan Wang, Xiangyu Chen, Yurong You, Li Li, Bharath Hariharan, Mark Campbell, Kilian Weinberger, and Wei-Lun Chao. Train in Germany, test in the USA: Making 3D object detectors generalize. In *CVPR*, 2020. 65, 129

[256] Yuqi Wang, Yuntao Chen, and Zhaoxiang Zhang. FrustumFormer: Adaptive instance-aware resampling for multi-view 3D detection. In *CVPR*, 2023. 59

[257] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. DETR3D: 3D object detection from multi-view images via 3D-to-2D queries. In *CoRL*, 2021. 38, 155

[258] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 39

[259] Zitian Wang, Zehao Huang, Jiahui Fu, Naiyan Wang, and Si Liu. Object as Query: Lifting any 2D object detector to 3D detection. In *ICCV*, 2023. 59

[260] Zeyu Wang, Dingwen Li, Chenxu Luo, Cihang Xie, and Xiaodong Yang. DistillBEV: Boosting multi-camera 3D object detection with cross-modal knowledge distillation. In *ICCV*, 2023. 48, 65

[261] Zengran Wang, Chen Min, Zheng Ge, Yinhao Li, Zeming Li, Hongyu Yang, and Di Huang. STS: Surround-view temporal stereo for multi-view 3D detection. In *AAAI*, 2023. 48, 65, 155

[262] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. Coordinate independent convolutional networks–isometry and gauge equivariant convolutions on riemannian manifolds. *arXiv preprint arXiv:2106.06020*, 2021. 28

[263] Maurice Weiler, Fred Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *CVPR*, 2018. 28

[264] Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. In *NeurIPS*, 2018. 28

[265] Daniel Worrall and Gabriel Brostow. Cubenet: Equivariance to 3D rotation and translation. In *ECCV*, 2018. 28, 29, 38

[266] Daniel Worrall, Stephan Garbin, Daniyar Turmukhambetov, and Gabriel Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017. 28

[267] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. In *NeurIPS*, 2019. 28, 39, 41, 121

[268] Chen Wu. Waymo keynote talk, CVPR workshop on autonomous driving at 17:20. https://www.youtube.com/watch?v=fXsbI2VkHgc, 2023. Accessed: 2023-11-11. 2, 45

[269] Pengxiang Wu, Siheng Chen, and Dimitris Metaxas. MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *CVPR*, 2020. 9

[270] Yuxin Wu and Justin Johnson. Rethinking "batch" in batchnorm. *arXiv preprint arXiv:2105.07576*, 2021. 127

[271] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, 2017. 19

[272] Enze Xie, Zhiding Yu, Daquan Zhou, Jonah Philion, Anima Anandkumar, Sanja Fidler, Ping Luo, and Jose Alvarez. M$\hat{2}$BEV: Multi-camera joint 3D detection and segmentation with unified birds-eye view representation. *arXiv preprint arXiv:2204.05088*, 2022. 48, 58

[273] Kaixin Xiong, Shi Gong, Xiaoqing Ye, Xiao Tan, Ji Wan, Errui Ding, Jingdong Wang, and Xiang Bai. CAPE: Camera view position embedding for multi-view 3D object detection. In *CVPR*, 2023. 59, 155

[274] Chenfeng Xu, Huan Ling, Sanja Fidler, and Or Litany. 3Difftection: 3D object detection with geometry-aware diffusion features. In *CVPR*, 2024. 65

[275] Junkai Xu, Liang Peng, Haoran Cheng, Hao Li, Wei Qian, Ke Li, Wenxiao Wang, and Deng Cai. MonoNeRD: NeRF-like representations for monocular 3D object detection. In *ICCV*, 2023. 47, 65

[276] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *ICLR*, 2021. 63, 64

[277] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles Qi, and Dragomir Anguelov. SPG: Unsupervised domain adaptation for 3D object detection via semantic point generation. In *ICCV*, 2021. 65

[278] Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*, 2014. 28

[279] Longfei Yan, Pei Yan, Shengzhou Xiong, Xuanyu Xiang, and Yihua Tan. MonoCD: Monocular 3D object detection with complementary depths. In *CVPR*, 2024. 65

[280] Gengshan Yang and Deva Ramanan. Upgrading optical flow to 3D scene flow through optical expansion. In *CVPR*, 2020. 34

[281] Haitao Yang, Zaiwei Zhang, Xiangru Huang, Min Bai, Chen Song, Bo Sun, Li Erran Li, and Qixing Huang. LiDAR-based 3D object detection via hybrid 2D semantic scene generation. *arXiv preprint arXiv:2304.01519*, 2023. 48, 58, 59

[282] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. ST3D: Self-training for unsupervised domain adaptation on 3D object detection. In *CVPR*, 2021. 65

[283] Jiayu Yang, Enze Xie, Miaomiao Liu, and Jose Alvarez. Parametric depth based feature representation learning for object detection and segmentation in bird's-eye view. In *ICCV*, 2023. 59

[284] Xiaodong Yang, Zhuang Ma, Zhiyu Ji, and Zhe Ren. GEDepth: Ground embedding for monocular depth estimation. In *ICCV*, 2023. 67, 159, 160

[285] Yue Yao, Shengchao Yan, Daniel Goehring, Wolfram Burgard, and Joerg Reichardt. Improving out-of-distribution generalization of trajectory prediction for autonomous driving via polynomial representations. *arXiv preprint arXiv:2407.13431*, 2024. 65

[286] Shingo Yashima, Teppei Suzuki, Kohta Ishikawa, Ikuro Sato, and Rei Kawakami. Feature space particle inference for neural network ensembles. In *ICML*, 2022. 64

[287] Xiaoqing Ye, Liang Du, Yifeng Shi, Yingying Li, Xiao Tan, Jianfeng Feng, Errui Ding, and Shilei Wen. Monocular 3D object detection via feature domain adaptation. In *ECCV*, 2020. 19

[288] Raymond Yeh, Yuan-Ting Hu, and Alexander Schwing. Chirality nets for human pose regression. In *NeurIPS*, 2019. 28

[289] Jingru Yi, Pengxiang Wu, Bo Liu, Qiaoying Huang, Hui Qu, and Dimitris Metaxas. Oriented object detection in aerial images with box boundary-aware vectors. In *WACV*, 2021. 55

[290] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D object detection and tracking. In *CVPR*, 2021. 1, 47, 64

[291] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2015. 38, 39, 41, 121

[292] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018. 123

[293] Xiang Yu, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *RSS*, 2018. 1, 44, 61

[294] Syed Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for fast image restoration and enhancement. *TPAMI*, 2022. 153

[295] Arthur Zhang, Chaitanya Eranki, Christina Zhang, Raymond Hong, Pranav Kalyani, Lochana Kalyanaraman, Arsh Gamare, Maria Esteva, and Joydeep Biswas. Towards robust 3D robot perception in urban environments: The UT Campus Object Dataset (CODa). In *IROS*, 2023. 164

[296] Hao Zhang, Hongyang Li, Xingyu Liao, Feng Li, Shilong Liu, Lionel Ni, and Lei Zhang. DA-BEV: Depth aware BEV transformer for 3D object detection. *arXiv preprint*

arXiv:2302.13002, 2023. 48, 65

[297] Jason Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *ICLR*, 2024. 65

[298] Jianming Zhang, Stan Sclaroff, Zhe Lin, Xiaohui Shen, Brian Price, and Radomir Mech. Unconstrained salient object detection via proposal subset optimization. In *CVPR*, 2016. 9, 16, 17

[299] Jinqing Zhang, Yanan Zhang, Qingjie Liu, and Yunhong Wang. SA-BEV: Generating semantic-aware bird's-eye-view feature for multi-view 3D object detection. In *ICCV*, 2023. 59

[300] Renrui Zhang, Han Qiu, Tai Wang, Xuanzhuo Xu, Ziyu Guo, Yu Qiao, Peng Gao, and Hongsheng Li. MonoDETR: Depth-guided transformer for monocular 3D object detection. In *ICCV*, 2023. 55, 56, 57, 154, 157

[301] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. Objects are different: Flexible monocular 3D object detection. In *CVPR*, 2021. 25, 29, 35, 36, 48, 65, 132

[302] Yinmin Zhang, Xinzhu Ma, Shuai Yi, Jun Hou, Zhihui Wang, Wanli Ouyang, and Dan Xu. Learning geometry-guided depth via projective modeling for monocular 3D object detection. *arXiv preprint arXiv:2107.13931*, 2021. 26

[303] Yunpeng Zhang, Zheng Zhu, Wenzhao Zheng, Junjie Huang, Guan Huang, Jie Zhou, and Jiwen Lu. BEVerse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving. *arXiv preprint arXiv:2205.09743*, 2022. 45, 48, 50, 53, 54, 55, 58, 59, 60, 65, 70, 149, 150, 151, 155

[304] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. In *ICLR*, 2021. 28

[305] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *CVPR*, 2022. 48

[306] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 25

[307] Yunsong Zhou, Yuan He, Hongzi Zhu, Cheng Wang, Hongyang Li, and Qinhong Jiang. MonoEF: Extrinsic parameter free monocular 3D object detection. *TPAMI*, 2021. 26, 29, 35, 36, 48, 61, 65, 132

[308] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3D object detection. In *CVPR Workshop*, 2019. 2, 45, 59

[309] Wei Zhu, Qiang Qiu, Robert Calderbank, Guillermo Sapiro, and Xiuyuan Cheng. Scale-equivariant neural networks with decomposed convolutional filters. *arXiv preprint*

arXiv:1909.11193, 2019. 27, 28, 32, 132

[310] Zijian Zhu, Yichi Zhang, Hai Chen, Yinpeng Dong, Shu Zhao, Wenbo Ding, Jiachen Zhong, and Shibao Zheng. Understanding the robustness of 3D object detection with bird's-eye-view representations in autonomous driving. In *CVPR*, 2023. 1, 44, 61

[311] Zhuofan Zong, Dongzhi Jiang, Guanglu Song, Zeyue Xue, Jingyong Su, Hongsheng Li, and Yu Liu. Temporal enhanced training of multi-view 3D object detector via historical object prediction. In *ICCV*, 2023. 45, 48, 55, 59, 60, 65, 149, 150, 151, 155

[312] Zhikang Zou, Xiaoqing Ye, Liang Du, Xianhui Cheng, Xiao Tan, Li Zhang, Jianfeng Feng, Xiangyang Xue, and Errui Ding. The devil is in the task: Exploiting reciprocal appearance-localization features for monocular 3D object detection. In *ICCV*, 2021. 35, 36

[313] Philip Zwicke and Imre Kiss. A new implementation of the mellin transform and its application to radar classification of ships. *TPAMI*, 1983. 27, 28, 33, 39, 41

APPENDIX A

PUBLICATIONS

First-Author Publications.

A list of all first-authored peer-reviewed publications during the Ph.D. program listed in reverse chronological order.

- *Abhinav Kumar*, Yuliang Guo, Zhihao Zhang, Xinyu Huang, Liu Ren and Xiaoming Liu. "CHARM3R: Towards Camera Height Agnostic Monocular 3D Object Detector ", ICCV, 2025 (under review).
- *Abhinav Kumar*, Yuliang Guo, Xinyu Huang, Liu Ren and Xiaoming Liu. "SeaBird: Segmentation in Bird's View with Dice Loss Improves 3D Detection of Large Objects", CVPR, 2024.
- *Abhinav Kumar*, Garrick Brazil, Enrique Corona, Armin Parchami and Xiaoming Liu. "DEVIANT: Depth Equivariant Network for Monocular 3D Object Detection", ECCV, 2022.
- *Abhinav Kumar*, Garrick Brazil and Xiaoming Liu. "GrooMeD-NMS: Grouped Mathematically Differentiable NMS for Monocular 3D Object Detection", CVPR, 2021.
- *Abhinav Kumar**, Tim Marks*, Wenxuan Mou*, Ye Wang, Michael Jones, Anoop Cherian, Toshi Koike-Akino, Xiaoming Liu and Chen Feng. "LUVLi Face Alignment: Estimating Location, Uncertainty and Visibility Likelihood", CVPR, 2020.

Other Publications.

- Yunfei Long, *Abhinav Kumar*, Xiaoming Liu and Daniel Morris. "RICCARDO: Radar Hit Prediction and Convolution for Camera-Radar 3D Object Detection", CVPR, 2025.
- Yuliang Guo, *Abhinav Kumar*, Chen Zhao, Ruoyu Wang, Xinyu Huang, and Liu Ren. "SUP-NeRF: A Streamlined Unification of Pose Estimation and NeRF for Monocular 3D Object Reconstruction", ECCV, 2024.
- Shengjie Zhu, Girish Ganesan, *Abhinav Kumar* and Xiaoming Liu. "RePLAy: Remove Projective LiDAR Depthmap Artifacts via Exploiting Epipolar Geometry", ECCV, 2024.

- Shengjie Zhu, *Abhinav Kumar*, Masa Hu and Xiaoming Liu. “Tame a Wild Camera: In-the-Wild Monocular Camera Calibration", NeurIPS, 2023.
- Vishal Asnani, *Abhinav Kumar*, Sua You and Xiaoming Liu. “PrObeD: Proactive 2D Object Detection Wrapper", NeurIPS, 2023.
- Garrick Brazil, *Abhinav Kumar*, Julian Straub, Nikhila Ravi, Justin Johnson and Georgia Gkioxari, “Omni3D: A Large Benchmark and Model for 3D Object Detection in the Wild", CVPR, 2023.
- Yunfei Long, *Abhinav Kumar*, Daniel Morris, Xiaoming Liu, Marcos Castro and Punarjay Chakravarty. “RADIANT: radar Image Association NeTwork for 3D Object Detection", AAAI, 2023.
- Thiago Serra, Xin Yu, *Abhinav Kumar*, and Srikumar Ramalingam. “Scaling Up Exact Neural Network Compression by ReLU Stability", NeurIPS, 2021.
- *Abhinav Kumar*[*], Tim Marks[*], Wenxuan Mou[*], Chen Feng and Xiaoming Liu. “UGLLI Face Alignment: Estimating Uncertainty with Gaussian Log-Likelihood Loss". ICCV Workshops, 2019.

APPENDIX B

GROOMED-NMS APPENDIX

B.1 Detailed Explanation of NMS as a Matrix Operation

The rescoring process of the classical NMS is greedy set-based [192] and calculates the rescore for a box i (Line 10 of Alg. 1) as

$$r_i = s_i \prod_{j \in \mathbf{d}_{<i}} \left(1 - p(o_{ij})\right), \tag{B.1}$$

where $\mathbf{d}_{<i}$ is defined as the box indices sampled from $\mathbf{d}$ having higher scores than box i. For example, let us consider that $\mathbf{d} = \{1, 5, 7, 9\}$. Then, for $i = 7$, $\mathbf{d}_{<i} = \{1, 5\}$ while for $i = 1, \mathbf{d}_{<i} = \phi$ with ϕ denoting the empty set. This is possible since we had sorted the scores $\mathbf{s}$ and $\mathbf{O}$ in decreasing order (Lines 2-3 of Alg. 2) to remove the non-differentiable hard argmax operation of the classical NMS (Line 6 of Alg. 1).

Classical NMS only takes the overlap with unsuppressed boxes into account. Therefore, we generalize Eq. (B.1) by accounting for the effect of all (suppressed and unsuppressed) boxes as

$$r_i = s_i \prod_{j=1}^{i-1} \left(1 - p(o_{ij}) r_j\right). \tag{B.2}$$

The presence of r_j on the RHS of Eq. (B.2) prevents suppressed boxes $r_j \approx 0$ from influencing other boxes hugely. Let us say we have a box b_2 with a high overlap with an unsuppressed box b_1. The classical NMS with a threshold pruning function assigns $r_2 = 0$ while Eq. (B.2) assigns r_2 a small non-zero value with a threshold pruning.

Although Eq. (B.2) keeps $r_i \geq 0$, getting a closed-form recursion in $\mathbf{r}$ is not easy because of the product operation. To get a closed-form recursion with addition/subtraction in $\mathbf{r}$, we first carry out the polynomial multiplication and then ignore the higher-order terms as

$$\begin{aligned} r_i &= s_i \left(1 - \sum_{j=1}^{i-1} p(o_{ij}) r_j + O(n^2)\right) \\ &\approx s_i \left(1 - \sum_{j=1}^{i-1} p(o_{ij}) r_j\right) \end{aligned}$$

Table B.1 **Results on using Oracle NMS scores on KITTI Val 1 cars detection**. [Key: **Best**]

NMS Scores	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)			AP $_{2D\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Kinematic (Image)	18.29	13.55	10.13	25.72	18.82	14.48	93.69	84.07	67.14
Oracle IoU_{2D}	9.36	9.93	6.40	12.27	10.43	8.72	**99.18**	**95.66**	**85.77**
Oracle IoU_{3D}	**87.93**	**73.10**	**60.91**	**93.47**	**83.61**	**71.31**	80.99	78.38	67.66

$$\approx s_i - \sum_{j=1}^{i-1} p(o_{ij}) r_j. \tag{B.3}$$

Dropping the s_i in the second term of Eq. (B.3) helps us get a cleaner form of Eq. (B.7). Moreover, it does not change the nature of the NMS since the subtraction keeps the relation $r_i \leq s_i$ intact as $p(o_{ij})$ and r_j are both between $[0, 1]$.

We can also reach Eq. (B.3) directly as follows. Classical NMS suppresses a box which has a high IoU_{2D} overlap with *any* of the unsuppressed boxes ($r_j \approx 1$) to zero. We consider *any* as a logical non-differentiable OR operation and use logical OR $\bigvee$ operator's differentiable relaxation as $\sum$ [106, 124]. We next use this relaxation with the other expression $\mathbf{r} \leq \mathbf{s}$.

When a box shows overlap with more than two unsuppressed boxes, the term $\sum_{j=1}^{i-1} p(o_{ij}) r_j > 1$ in Eq. (B.3) or when a box shows high overlap with one unsuppressed box, the term $s_i < p(o_{ij}) r_j$. In both of these cases, $r_i < 0$. So, we lower bound Eq. (B.3) with a max operation to ensure that $r_i \geq 0$. Thus,

$$r_i \approx \max\left(s_i - \sum_{j=1}^{i-1} p(o_{ij}) r_j,\ 0 \right). \tag{B.4}$$

We write the rescores $\mathbf{r}$ in a matrix formulation as

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} \approx \max\left(\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix} - \begin{bmatrix} 0 & 0 & \dots & 0 \\ p(o_{21}) & 0 & \dots & 0 \\ p(o_{31}) & p(o_{32}) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ p(o_{n1}) & p(o_{n2}) & \dots & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right). \tag{B.5}$$

We next write the above equation compactly as

$$\mathbf{r} \approx \max(\mathbf{s} - \mathbf{P}\mathbf{r}, \mathbf{0}), \tag{B.6}$$

where $\mathbf{P}$, called the Prune Matrix, is obtained by element-wise operation of the pruning function p on $\mathbf{O}_{\llcorner}$. Maximum operation makes Eq. (B.6) non-linear [112] and, thus, difficult to solve.

However, for a differentiable NMS layer, we need to avoid the recursion. Therefore, we first solve Eq. (B.6) assuming the max operation is not present which gives us the solution $\mathbf{r} \approx (\mathbf{I} + \mathbf{P})^{-1}\,\mathbf{s}$. In general, this solution is not necessarily bounded between 0 and 1. Hence, we clip it explicitly to obtain the approximation

$$\mathbf{r} \approx \left\lfloor (\mathbf{I} + \mathbf{P})^{-1}\,\mathbf{s} \right\rceil, \tag{B.7}$$

which we use as the solution to Eq. (B.6).

B.2 Loss Functions

We now detail out the loss functions used for training. The losses on the boxes before NMS, $\mathcal{L}_{before}$, is given by [17]

$$\begin{aligned} \mathcal{L}_{before} = \mathcal{L}_{class} + \mathcal{L}_{\text{2D}} + b_{conf}\ \mathcal{L}_{\text{3D}} \\ + \lambda_{conf}(1 - b_{conf}), \end{aligned} \tag{B.8}$$

where

$$\mathcal{L}_{class} = \text{CE}(b_{class}, g_{class}), \tag{B.9}$$

$$\mathcal{L}_{\text{2D}} = -\log(\text{IoU}(b_{\text{2D}}, g_{\text{2D}})), \tag{B.10}$$

$$\begin{aligned} \mathcal{L}_{\text{3D}} = \text{Smooth-L1}(b_{\text{3D}}, g_{\text{3D}}) \\ + \lambda_a \text{CE}([b_{\theta_a}, b_{\theta_h}], [g_{\theta_a}, g_{\theta_h}]). \end{aligned} \tag{B.11}$$

b_{conf} is the predicted self-balancing confidence of each box b, while b_{θ_a} and b_{θ_h} are its orientation bins [17]. g denotes the ground-truth. λ_{conf} is the rolling mean of most recent $\mathcal{L}_{\text{3D}}$ losses per mini-batch [17], while λ_a denotes the weight of the orientation bins loss. CE and Smoooth-L1 denote the Cross Entropy and Smooth L1 loss respectively. Note that we apply 2D and 3D regression losses as well as the confidence losses only on the foreground boxes.

Table B.2 **Detailed comparisons with other NMS** during inference on KITTI Val 1 cars.

	Inference NMS	$\text{IoU}_{3D} \geq 0.7$						$\text{IoU}_{3D} \geq 0.5$					
		$\text{AP}_{3D\|R_{40}}$ (↑)			$\text{AP}_{BEV\|R_{40}}$ (↑)			$\text{AP}_{3D\|R_{40}}$ (↑)			$\text{AP}_{BEV\|R_{40}}$ (↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Kinematic (Image) [17]	Classical	18.28	13.55	10.13	25.72	18.82	14.48	54.70	39.33	31.25	60.87	44.36	34.48
Kinematic (Image) [17]	Soft [12]	18.29	13.55	10.13	25.71	18.81	14.48	54.70	39.33	31.26	60.87	44.36	34.48
Kinematic (Image) [17]	Distance [216]	18.25	13.53	10.11	25.71	18.82	14.48	54.70	39.33	31.26	60.87	44.36	34.48
Kinematic (Image) [17]	GrooMeD	18.26	13.51	10.10	25.67	18.77	14.44	54.59	39.25	31.18	60.78	44.28	34.40
GrooMeD-NMS	Classical	19.67	14.31	11.27	27.38	19.75	15.93	55.64	41.08	32.91	61.85	44.98	36.31
GrooMeD-NMS	Soft [12]	19.67	14.31	11.27	27.38	19.75	15.93	55.64	41.08	32.91	61.85	44.98	36.31
GrooMeD-NMS	Distance [216]	19.67	14.31	11.27	27.38	19.75	15.93	55.64	41.08	32.91	61.85	44.98	36.31
GrooMeD-NMS	GrooMeD	19.67	14.32	11.27	27.38	19.75	15.92	55.62	41.07	32.89	61.83	44.98	36.29

As explained in Sec. 2.4.3, the loss on the boxes after NMS, $\mathcal{L}_{after}$, is the Imagewise AP-Loss, which is given by

$$\mathcal{L}_{after} = \mathcal{L}_{Imagewise} = \frac{1}{N}\sum_{m=1}^{N} \text{AP}(\mathbf{r}^{(m)}, \text{target}(\mathcal{B}^{(m)})), \tag{B.12}$$

Let λ be the weight of the $\mathcal{L}_{after}$ term. Then, our overall loss function is given by

$$\mathcal{L} = \mathcal{L}_{before} + \lambda \mathcal{L}_{after} \tag{B.13}$$

$$= \mathcal{L}_{class} + \mathcal{L}_{\text{2D}} + b_{conf}\ \mathcal{L}_{\text{3D}} + \lambda_{conf}(1 - b_{conf}) + \lambda \mathcal{L}_{Imagewise} \tag{B.14}$$

$$= \text{CE}(b_{class}, g_{class}) - \log(\text{IoU}(b_{\text{2D}}, g_{\text{2D}})) + b_{conf}\ \text{Smooth-L1}(b_{\text{3D}}, g_{\text{3D}}) + \lambda_a\ b_{conf}\ \text{CE}([b_{\theta_a}, b_{\theta_h}], [g_{\theta_a}, g_{\theta_h}]) + \lambda_{conf}(1 - b_{conf}) + \lambda \mathcal{L}_{Imagewise}. \tag{B.15}$$

We keep $\lambda_a = 0.35$ following [17] and $\lambda = 0.05$. Clearly, all our losses and their weights are identical to [17] except $\mathcal{L}_{Imagewise}$.

B.3 Additional Experiments and Results

We now provide additional details and results evaluating our system's performance.

B.3.1 Training

Training images are augmented using random flipping with probability 0.5 [17]. Adam optimizer [102] is used with batch size 2, weight-decay 5×10^{-4} and gradient clipping of 1 [15, 17].

Warmup starts with a learning rate 4×10^{-3} following a poly learning policy with power 0.9 [17]. Warmup and full training phases take $80k$ and $50k$ mini-batches respectively for Val 1 and Val 2 Splits [17] while take $160k$ and $100k$ mini-batches for Test Split.

B.3.2 KITTI Val 1 Oracle NMS Experiments

As discussed in Sec. 2.1, to understand the effects of an inference-only NMS on 2D and 3D object detection, we conduct a series of oracle experiments. We create an oracle NMS by taking the Val Car boxes of KITTI Val 1 Split from the baseline Kinematic (Image) model *before* NMS and replace their scores with their true IoU_{2D} or IoU_{3D} with the ground-truth, respectively. Note that this corresponds to the oracle because we do not know the ground-truth boxes during inference. We then pass the boxes with the oracle scores through the classical NMS and report the results in Tab. B.1.

The results show that the AP_{3D} increases by a staggering > 60 AP on Mod cars when we use oracle IoU_{3D} as the NMS score. On the other hand, we only see an increase in AP_{2D} by ≈ 11 AP on Mod cars when we use oracle IoU_{2D} as the NMS score. Thus, the relative effect of using oracle IoU_{3D} NMS scores on 3D detection is more significant than using oracle IoU_{2D} NMS scores on 2D detection. In other words, the mismatch is greater between classification and 3D localization compared to the mismatch between classification and 2D localization.

B.3.3 KITTI Val 1 3D Object Detection

Comparisons with other NMS. We compare GrooMeD-NMS with the other NMS—classical, Soft [12] and Distance-NMS [216] and report the detailed results in Tab. B.2. We use the publicly released Soft-NMS code and Distance-NMS code from the respective authors. The Distance-NMS model uses the class confidence scores divided by the uncertainty in z (the most erroneous dimension in 3D localization [221]) of a box as the Distance-NMS [216] input. Our model does not predict the uncertainty in z of a box but predicts its self-balancing confidence (the 3D localization score). Therefore, we use the class confidence scores multiplied by the self-balancing confidence as the Distance-NMS input.

The results in Tab. B.2 show that NMS inclusion in the training pipeline benefits the perfor-

Table B.3 **Sensitivity to NMS threshold** N_t on KITTI Val 1 cars. [Key: **Best**]

	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard
$N_t = 0.3$	17.49	13.32	10.54	26.07	18.94	14.61
N_t =**0.4**	**19.67**	**14.32**	**11.27**	**27.38**	**19.75**	**15.92**
$N_t = 0.5$	19.65	13.93	11.09	26.15	19.15	14.71

Table B.4 **Sensitivity to valid box threshold** v on KITTI Val 1 cars. [Key: **Best**]

	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard
$v = 0.01$	13.71	9.65	7.24	17.73	12.47	9.36
$v = 0.1$	19.37	13.99	10.92	26.95	19.84	15.40
$v = 0.2$	19.65	14.31	11.24	27.35	19.73	15.89
$v = 0.3$	19.67	14.32	11.27	27.38	19.75	15.92
$v = 0.4$	19.67	14.33	11.28	27.38	19.76	15.93
$v = 0.5$	19.67	14.33	11.28	27.38	19.76	15.93
v = **0.6**	**19.67**	**14.33**	**11.29**	**27.39**	**19.77**	**15.95**

mance, unlike [12], which suggests otherwise. Training with GrooMeD-NMS helps because the network gets an additional signal through the GrooMeD-NMS layer whenever the best-localized box corresponding to an object is not selected. Moreover, Tab. B.2 suggests that we can replace GrooMeD-NMS with the classical NMS in inference as the performance is almost the same even at $\text{IoU}_{3D} = 0.5$.

How good is the classical NMS approximation? GrooMeD-NMS uses several approximations to arrive at the matrix solution Eq. (B.7). We now compare how good these approximations are with the classical NMS. Interestingly, Tab. B.2 shows that GrooMeD-NMS is an excellent approximation to the classical NMS as the performance does not degrade after changing the NMS in inference.

B.3.4 KITTI Val 1 Sensitivity Analysis

There are a few adjustable parameters for the GrooMeD-NMS, such as the NMS threshold N_t, valid box threshold v, the maximum group size α, the weight λ for the $\mathcal{L}_{after}$, and β. We carry out a sensitivity analysis to understand how these parameters affect performance and speed, and how sensitive the algorithm is to these parameters.

Sensitivity to NMS Threshold. We show the sensitivity to NMS threshold N_t in Tab. B.3. The results in Tab. B.3 show that the optimal $N_t = 0.4$. This is also the N_t in [15, 17].

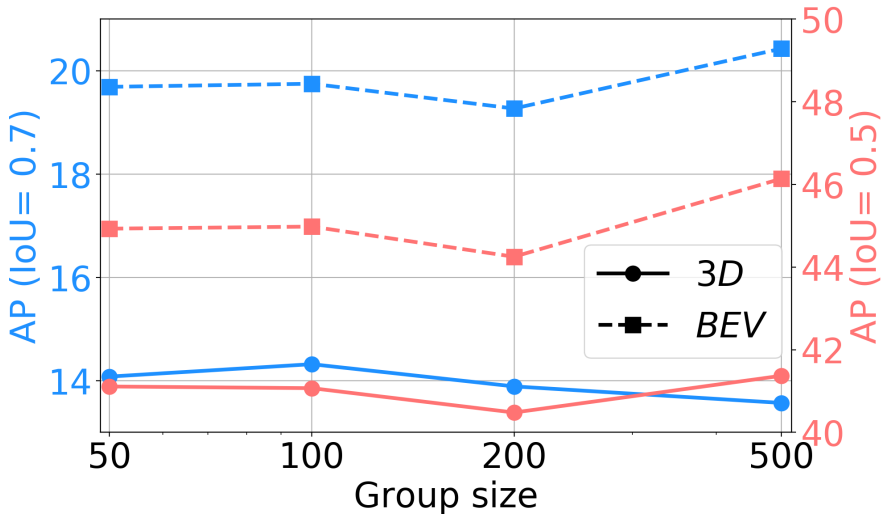


Figure B.1 **Sensitivity to group size** α on KITTI Val 1 Moderate cars.

Sensitivity to Valid Box Threshold. We next show the sensitivity to valid box threshold v in Tab. B.4. Our choice of $v = 0.3$ performs close to the optimal choice.

Sensitivity to Maximum Group Size. Grouping has a parameter group size (α). We vary this parameter and report AP $_{3D|R_{40}}$ and AP $_{BEV|R_{40}}$ at two different IoU$_{3D}$ thresholds on Moderate cars of KITTI Val 1 Split in Fig. B.1. We note that the best AP $_{3D|R_{40}}$ performance is obtained at $\alpha = 100$ and we, therefore, set $\alpha = 100$ in our experiments.

Sensitivity to Loss Weight. We now show the sensitivity to loss weight λ in Tab. B.5. Our choice of $\lambda = 0.05$ is the optimal value.

Sensitivity to Best Box Threshold. We now show the sensitivity to the best box threshold β in Tab. B.6. Our choice of $\beta = 0.3$ is the optimal value.

Conclusion. GrooMeD-NMS has minor sensitivity to N_t, α, λ and β, which is common in object detection. GrooMeD-NMS is not as sensitive to v since it only decides a box's validity. Our parameter choice is either at or close to the optimal. The inference speed is only affected by α. Other parameters are used in training or do not affect inference speed.

B.3.5 Qualitative Results

We next show some qualitative results of models trained on KITTI Val 1 Split in Fig. B.2. We depict the predictions of GrooMeD-NMS in image view on the left and the predictions of GrooMeD-NMS, Kinematic (Image) [17], and ground truth in BEV on the right. In general,

Table B.5 **Sensitivity to loss weight** λ on KITTI Val 1 cars. [Key: **Best**]

	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard
$\lambda = 0$	19.16	13.89	10.96	27.01	19.33	14.84
$\lambda = \mathbf{0.05}$	**19.67**	**14.32**	**11.27**	**27.38**	**19.75**	**15.92**
$\lambda = 0.1$	17.74	13.61	10.81	25.86	19.18	15.57
$\lambda = 1$	10.08	7.26	6.00	14.44	10.55	8.41

Table B.6 **Sensitivity to best box threshold** β on KITTI Val 1 cars. [Key: **Best**]

	AP $_{3D\|R_{40}}$ (↑)			AP $_{BEV\|R_{40}}$ (↑)		
	Easy	Mod	Hard	Easy	Mod	Hard
$\beta = 0.1$	18.09	13.64	10.21	26.52	19.50	15.74
$\beta = \mathbf{0.3}$	**19.67**	**14.32**	**11.27**	**27.38**	**19.75**	**15.92**
$\beta = 0.4$	18.91	14.02	11.15	27.11	19.64	15.90
$\beta = 0.5$	18.49	13.66	10.96	27.01	19.47	15.79

GrooMeD-NMS predictions are more closer to the ground truth than Kinematic (Image) [17].

B.3.6 Demo Video of GrooMeD-NMS

We next include a short demo video of our GrooMeD-NMS model trained on KITTI Val 1 Split. We run our trained model independently on each frame of the three KITTI raw [66] sequences - 2011_10_03_drive_0047, 2011_09_29_drive_0026 and 2011_09_26_drive_0009. None of the frames from these three raw sequences appear in the training set of KITTI Val 1 Split. We use the camera matrices available with the raw sequences but do not use any temporal information. Overlaid on each frame of the raw input videos, we plot the projected 3D boxes of the predictions and also plot these 3D boxes in the BEV. We set the frame rate of this demo at 10 fps. The demo is also available in HD at https://www.youtube.com/watch?v=PWctKkyWrno. In the demo video, notice that the orientation of the boxes are stable despite not using any temporal information.

Figure B.2 **Qualitative Results** (Best viewed in color). We depict the predictions of GrooMeD-NMS (magenta) in image view on the left and the predictions of GrooMeD-NMS, Kinematic (Image) [17] (blue), and Ground Truth (green) in BEV on the right. In general, GrooMeD-NMS predictions are more closer to the ground truth than Kinematic (Image) [17].

APPENDIX C

DEVIANT APPENDIX

C.1 Supportive Explanations

We now add some explanations which we could not put in the main chapter because of the space constraints.

C.1.1 Equivariance vs Augmentation

Equivariance adds suitable inductive bias to the backbone [43, 50] and is not learnt. Augmentation adds transformations to the input data during training or inference.

Equivariance and data augmentation have their own pros and cons. Equivariance models the physics better, is mathematically principled and is so more agnostic to data distribution shift compared to the data augmentation. A downside of equivariance compared to the augmentation is equivariance requires mathematical modelling, may not always exist [21], is not so intuitive and generally requires more flops for inference. On the other hand, data augmentation is simple, intuitive and fast, but is not mathematically principled. The choice between equivariance and data augmentation is a withstanding question in machine learning [63].

C.1.2 Why do 2D CNN detectors generalize?

We now try to understand why 2D CNN detectors generalize well. Consider an image $h(u, v)$ and Φ be the CNN. Let $\mathcal{T}_{\mathbf{t}}$ denote the translation in the (u, v) space. The 2D translation equivariance [19, 20, 198] of the CNN means that

$$\begin{aligned} &\Phi(\mathcal{T}_{\mathbf{t}} h(u, v)) = \mathcal{T}_{\mathbf{t}} \Phi(h(u, v)) \\ \implies &\Phi(h(u + t_u, v + t_v)) = \Phi(h(u, v)) + (t_u, t_v) \end{aligned} \tag{C.1}$$

where (t_u, t_v) is the translation in the (u, v) space.

Assume the CNN predicts the object position in the image as (u', v'). Then, we write

$$\Phi(h(u, v)) = (\hat{u}, \hat{v}) \tag{C.2}$$

Now, we want the CNN to predict the output the position of the same object translated by (t_u, t_v). The new image is thus $h(u + t_u, v + t_v)$. The CNN easily predicts the translated position of

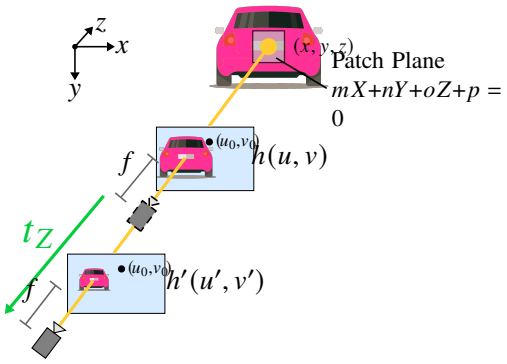


Figure C.1 **Equivariance exists** for the patch plane when there is depth translation of the ego camera. Downscaling converts image h to image h'.

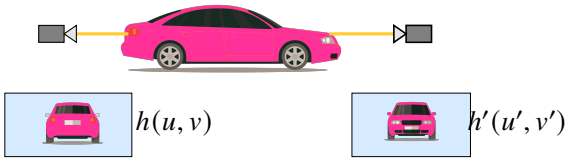


Figure C.2 **Example of non-existence of equivariance** [21] when there is 180° rotation of the ego camera. No transformation can convert image h to image h'.

the object because all CNN is to do is to invoke its 2D translation equivariance of Eq. (C.1), and translate the previous prediction by the same amount. In other words,

$$\begin{aligned}\Phi(h(u+t_u, v+t_v)) &= \Phi(h(u,v)) + (t_u, t_v) \\ &= (\hat{u}, \hat{v}) + (t_u, t_v) \\ &= (\hat{u}+t_u, \hat{v}+t_v)\end{aligned}$$

Intuitively, equivariance is a disentaglement method. The 2D translation equivariance disentangles the 2D translations (t_u, t_v) from the original image h and therefore, the network generalizes to unseen 2D translations.

C.1.3 Existence and Non-existence of Equivariance

The result from [21] says that generic projective equivariance does not exist in particular with rotation transformations. We now show an example of when the equivariance exists and does not exist in the projective manifold in Figs. C.1 and C.2 respectively.

C.1.4 Why do not Monocular 3D CNN detectors generalize?

Monocular 3D CNN detectors do not generalize well because they are not equivariant to arbitrary 3D translations in the projective manifold. To show this, let $H(X, Y, Z)$ denote a 3D point cloud. The monocular detection network Φ operates on the projection $h(u, v)$ of this point cloud H to output the position $(\hat{x}, \hat{y}, \hat{z})$ as

$$\begin{aligned} &\Phi(\mathbf{K}H(X, Y, Z)) = (\hat{x}, \hat{y}, \hat{z}) \\ \implies &\Phi(h(u, v)) = (\hat{x}, \hat{y}, \hat{z}), \end{aligned}$$

where $\mathbf{K}$ denotes the projection operator. We translate this point cloud by an arbitrary 3D translation of (t_X, t_Y, t_Z) to obtain the new point cloud $H(X + t_X, Y + t_Y, Z + t_Z)$. Then, we again ask the monocular detector Φ to do prediction over the translated point cloud. However, we find that

$$\begin{aligned} \Phi(\mathbf{K}H(X + t_X, Y + t_Y, Z + t_Z)) &\neq \Phi(h(u + \mathbf{K}(t_X, t_Y, t_Z), v + \mathbf{K}(t_X, t_Y, t_Z))) \\ &= \Phi(h(u, v)) + \mathbf{K}(t_X, t_Y, t_Z) \\ \implies \Phi(\mathbf{K}H(X + t_X, Y + t_Y, Z + t_Z)) &\neq \Phi(\mathbf{K}H(X, Y, Z)) + \mathbf{K}(t_X, t_Y, t_Z) \end{aligned}$$

In other words, the projection operator $\mathbf{K}$ does not distribute over the point cloud H and arbitrary 3D translation of (t_X, t_Y, t_Z). Hence, if the network Φ is a vanilla CNN (existing monocular backbone), it can no longer invoke its 2D translation equivariance of Eq. (C.1) to get the new 3D coordinates $(\hat{x} + t_X, \hat{y} + t_Y, \hat{z} + t_Z)$.

Note that the LiDAR based 3D detectors with 3D convolutions do not suffer from this problem because they do not involve any projection operator $\mathbf{K}$. Thus, this problem exists only in monocular 3D detection. This makes monocular 3D detection different from 2D and LiDAR based 3D object detection.

C.1.5 Overview of Planar Transformations: Th. 1

We now pictorially provide the overview of Th. 1 (Example 13.2 from [76]), which links the planarity and projective transformations in the continuous world in Fig. C.3.

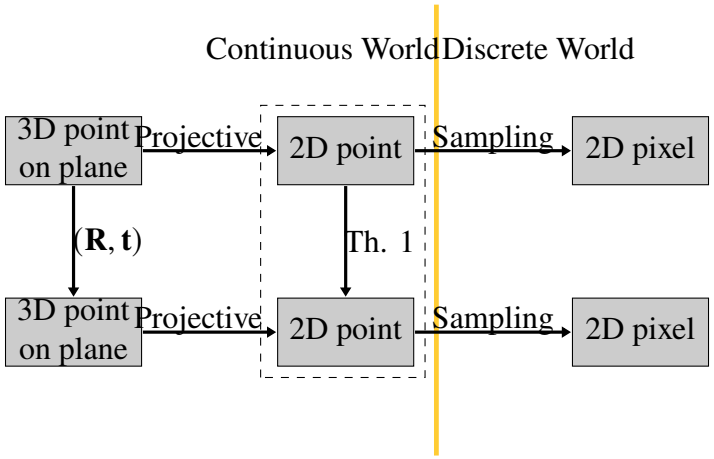


Figure C.3 **Overview** of Th. 1 (Example 13.2 from [76]), which links the planarity and projective transformations in the continuous world.

C.1.6 Approximation of Scale Transformations: Corollary 1.1

We now give the approximation under which Corollary 1.1 is valid. We assume that the ego camera does not undergo any rotation. Hence, we substitute $\mathbf{R} = \mathbf{I}$ in Eq. (3.1) to get

$$h(u-u_0, v-v_0) = h'\left(f\frac{\left(1+t_X\frac{m}{p}\right)(u-u_0)+t_X\frac{n}{p}(v-v_0)+t_X\frac{o}{p}f}{t_Z\frac{m}{p}(u-u_0)+t_Z\frac{n}{p}(v-v_0)+\left(1+t_Z\frac{o}{p}\right)f}, \right.$$
$$\left. f\frac{t_Y\frac{m}{p}(u-u_0)+\left(1+t_Y\frac{n}{p}\right)(v-v_0)+t_Y\frac{o}{p}f}{t_Z\frac{m}{p}(u-u_0)+t_Z\frac{n}{p}(v-v_0)+\left(1+t_Z\frac{o}{p}\right)f} \right). \tag{C.3}$$

Next, we use the assumption that the ego vehicle moves in the z-direction as in [17], *i.e.*, substitute $t_X = t_Y = 0$ to get

$$h(u-u_0, v-v_0) = h'\left(\frac{u-u_0}{\frac{t_Z}{f}\frac{m}{p}(u-u_0)+\frac{t_Z}{f}\frac{n}{p}(v-v_0)+\left(1+t_Z\frac{o}{p}\right)}, \right.$$
$$\left. \frac{v-v_0}{\frac{t_Z}{f}\frac{m}{p}(u-u_0)+\frac{t_Z}{f}\frac{n}{p}(v-v_0)+\left(1+t_Z\frac{o}{p}\right)} \right). \tag{C.4}$$

The patch plane is $mx+ny+oz+p=0$. We consider the planes in the front of camera. Without loss of generality, consider $p<0$ and $o>0$.

We first write the denominator D of RHS term in Eq. (C.4) as

$$D = \frac{t_Z}{f}\frac{m}{p}(u-u_0)+\frac{t_Z}{f}\frac{n}{p}(v-v_0)+\left(1+t_Z\frac{o}{p}\right)$$

$$= 1 + \frac{t_Z}{p}\left(\frac{m}{f}(u-u_0) + \frac{n}{f}(v-v_0) + o\right)$$

Because we considered patch plane s in front of the camera, $p < 0$. Also consider $t_Z < 0$, which implies $t_Z/p > 0$. Now, we bound the term in the parantheses of the above equation as

$$\begin{aligned}
D &\leq 1 + \frac{t_Z}{p}\left\|\frac{m}{f}(u-u_0) + \frac{n}{f}(v-v_0) + o\right\| \\
&\leq 1 + \frac{t_Z}{p}\left(\left\|\frac{m}{f}(u-u_0)\right\| + \left\|\frac{n}{f}(v-v_0)\right\| + \|o\|\right) \quad \text{by Triangle inequality} \\
&\leq 1 + \frac{t_Z}{p}\left(\frac{\|m\|}{f}\frac{W}{2} + \frac{\|n\|}{f}\frac{H}{2} + o\right), (u-u_0) \leq \frac{W}{2}, (v-v_0) \leq \frac{H}{2}, \|o\| = o \\
&\leq 1 + \frac{t_Z}{p}\left(\frac{\|m\|}{f}\frac{W}{2} + \frac{\|n\|}{f}\frac{W}{2} + o\right), H \leq W \\
&\leq 1 + \frac{t_Z}{p}\left(\frac{(\|m\| + \|n\|)W}{2f} + o\right),
\end{aligned}$$

If the coefficients of the patch plane m, n, o, its width W and focal length f follow the relationship $\frac{(\|m\|+\|n\|)W}{2f} << o$, the patch plane is "approximately" parallel to the image plane. Then, a few quantities can be ignored in the denominator D to get

$$\mathrm{D} \approx 1 + t_Z\frac{o}{p} \tag{C.5}$$

Therefore, the RHS of Eq. (C.4) gets simplified and we obtain

$$\mathcal{T}_s : h(u-u_0, v-v_0) \approx h'\left(\frac{u-u_0}{1+t_Z\frac{o}{p}}, \frac{v-v_0}{1+t_Z\frac{o}{p}}\right) \tag{C.6}$$

An immediate benefit of using the approximation is Eq. (3.2) does not depend on the distance of the patch plane from the camera. This is different from wide-angle camera assumption, where the ego camera is assumed to be far from the patch plane. Moreover, patch plane s need not be perfectly aligned with the image plane for Eq. (3.2). Even small enough perturbed patch plane s work. We next show the approximation in the Fig. C.4 with θ denoting the deviation from the perfect parallel plane. The deviation θ is about 3 degrees for the KITTI dataset while it is 6 degrees for the Waymo dataset.

e.g. The following are valid patch plane s for KITTI images whose focal length $f = 707$ and width $W = 1242$.

$$-0.05x + 0.05y + z = 30$$

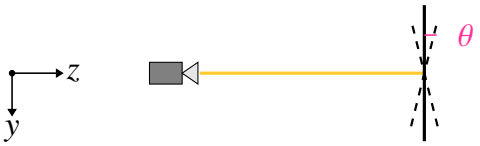


Figure C.4 **Approximation of Corollary 1.1**. Bold shows the patch plane parallel to the image plane. The dotted line shows the approximated patch plane.

$$0.05x - 0.05y + z = 30 \tag{C.7}$$

The following are valid patch plane s for Waymo images whose focal length $f = 2059$ and width $W = 1920$.

$$\begin{aligned} -0.1x + 0.1y + z &= 30 \\ 0.1x - 0.1y + z &= 30 \end{aligned} \tag{C.8}$$

Although the assumption is slightly restrictive, we believe our method shows improvements on both KITTI and Waymo datasets because the car patches are approximately parallel to image planes and also because the depth remains the hardest parameter to estimate [166].

C.1.7 Scale Equivariance of SES Convolution for Images

[227] derive the scale equivariance of SES convolution for a 1D signal. We simply follow on their footsteps to get the scale equivariance of SES convolution for a 2D image $h(u, v)$ for the sake of completeness. Let the scaling of the image h be s. Let $*$ denote the standard vanilla convolution and Ψ denote the convolution filter. Then, the convolution of the downscaled image $\mathcal{T}_s(h)$ with the filter Ψ is given by

$$\begin{aligned} &[\mathcal{T}_s(h) * \Psi]\,(u, v) \\ &= \int\int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \Psi(u' - u, v' - v) du' dv' \\ &= s^2 \int\int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \Psi\left(s\frac{u' - u}{s}, s\frac{v' - v}{s}\right) d\left(\frac{u'}{s}\right) d\left(\frac{v'}{s}\right) \\ &= s^2 \int\int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \mathcal{T}_{s^{-1}}\left[\Psi\left(\frac{u' - u}{s}, \frac{v' - v}{s}\right)\right] d\left(\frac{u'}{s}\right) d\left(\frac{v'}{s}\right) \\ &= s^2 \int\int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \mathcal{T}_{s^{-1}}\left[\Psi\left(\frac{u'}{s} - \frac{u}{s}, \frac{v'}{s} - \frac{v}{s}\right)\right] d\left(\frac{u'}{s}\right) d\left(\frac{v'}{s}\right) \end{aligned}$$

$$= s^2 \left[h * \mathcal{T}_{s^{-1}}(\Psi)\right] \left(\frac{u}{s}, \frac{v}{s}\right)$$
$$= s^2 \mathcal{T}_s \left[h * \mathcal{T}_{s^{-1}}(\Psi)\right](u, v). \tag{C.9}$$

Next, [227] re-parametrize the SES filters by writing $\Psi_\sigma(u, v) = \frac{1}{\sigma^2}\Psi\left(\frac{u}{\sigma}, \frac{v}{\sigma}\right)$. Substituting in Eq. (C.9), we get

$$\left[\mathcal{T}_s(h) * \Psi_\sigma\right](u, v) = s^2 \mathcal{T}_s \left[h * \mathcal{T}_{s^{-1}}(\Psi_\sigma)\right](u, v) \tag{C.10}$$

Moreover, the re-parametrized filters are separable [227] by construction and so, one can write

$$\Psi_\sigma(u, v) = \Psi_\sigma(u)\Psi_\sigma(v). \tag{C.11}$$

The re-parametrization and separability leads to the important property that

$$\begin{aligned}
\mathcal{T}_{s^{-1}}\left(\Psi_\sigma(u, v)\right) &= \mathcal{T}_{s^{-1}}\left(\Psi_\sigma(u)\Psi_\sigma(v)\right) \\
&= \mathcal{T}_{s^{-1}}\left(\Psi_\sigma(u)\right) \mathcal{T}_{s^{-1}}\left(\Psi_\sigma(v)\right) \\
&= s^{-2}\Psi_{s^{-1}\sigma}(u)\Psi_{s^{-1}\sigma}(v) \\
&= s^{-2}\Psi_{s^{-1}\sigma}(u, v).
\end{aligned} \tag{C.12}$$

Substituting above in the RHS of Eq. (C.10), we get

$$\begin{aligned}
&\left[\mathcal{T}_s(h) * \Psi_\sigma\right](u, v) = s^2 \mathcal{T}_s \left[h * s^{-2}\Psi_{s^{-1}\sigma}\right](u, v) \\
\implies &\left[\mathcal{T}_s(h) * \Psi_\sigma\right](u, v) = \mathcal{T}_s \left[h * \Psi_{s^{-1}\sigma}\right](u, v),
\end{aligned} \tag{C.13}$$

which is a cleaner form of Eq. (C.9). Eq. (C.13) says that convolving the downscaled image with a filter is same as the downscaling the result of convolving the image with the upscaled filter [227]. This additional constraint regularizes the scale (depth) predictions for the image, leading to better generalization.

C.1.8 Why does DEVIANT generalize better compared to CNN backbone?

DEVIANT models the physics better compared to the CNN backbone. CNN generalizes better for 2D detection because of the 2D translation equivariance in the Euclidean manifold. However,

Table C.1 **Comparison of Methods** on the basis of inputs, convolution kernels, outputs and whether output are scale-constrained.

Method	Input Frame	#Conv Kernel	Output	Output Constrained for Scales?
Vanilla CNN	1	1	4D	×
Depth-Aware [15]	1	> 1	4D	×
Dilated CNN [291]	1	> 1	5D	Integer [267]
DEVIANT	1	> 1	5D	Float
Depth-guided [52]	1 + Depth	1	4D	Integer [267]
Kinematic3D [17]	> 1	1	5D	×

monocular 3D detection does not belong to the Euclidean manifold but is a task of the projective manifold. Modeling translation equivariance in the correct manifold improves generalization. For monocular 3D detection, we take the first step towards the general 3D translation equivariance by embedding equivariance to depth translations. The 3D depth equivariance in DEVIANT uses Eq. (C.10) and thus imposes an additional constraint on the feature maps. This additional constraint results in consistent depth estimates from the current image and a virtual image (obtained by translating the ego camera), and therefore, better generalization than CNNs. On the other hand, CNNs, by design, do not constrain the depth estimates from the current image and a virtual image (obtained by translating the ego camera), and thus, their depth estimates are entirely data-driven.

C.1.9 Why not Fixed Scale Assumption?

We now answer the question of keeping the fixed scale assumption. If we assume fixed scale assumption, then vanilla convolutional layers have the right equivariance. However, we do not keep this assumption because the ego camera translates along the depth in driving scenes and also, because the depth is the hardest parameter to estimate [166] for monocular detection. So, zero depth translation or fixed scale assumption is always violated.

C.1.10 Comparisons with Other Methods

We now list out the differences between different convolutions and monocular detection methods in Tab. C.1. Kinematic3D [17] does not constrain the output at feature map level, but at system level using Kalman Filters. The closest to our method is the Dilated CNN (DCNN) [291]. We show in Tab. 3.9 that DEVIANT outperforms Dilated CNN.

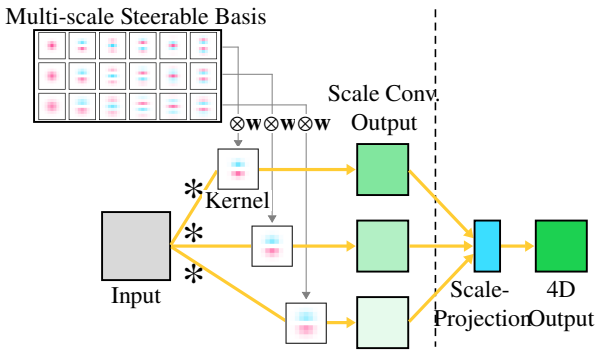


Figure C.5 **(a) SES convolution** [68, 227] The non-trainable basis functions multiply with learnable weights **w** to get kernels. The input then convolves with these kernels to get multi-scale 5D output. **(b) Scale-Projection** [227] takes max over the scale dimension of the 5D output and converts it to 4D. [Key: $*$ = Vanilla convolution.]

C.1.11 Why is Depth the hardest among all parameters?

Images are the 2D projections of the 3D scene, and therefore, the depth is lost during projection. Recovering this depth is the most difficult to estimate, as shown in Tab. 1 of [166]. Monocular detection task involves estimating 3D center, 3D dimensions and the yaw angle. The right half of Tab. 1 in [166] shows that if the ground truth 3D center is replaced with the predicted center, the detection reaches a minimum. Hence, 3D center is the most difficult to estimate among center, dimensions and pose. Most monocular 3D detectors further decompose the 3D center into projected (2D) center and depth. Out of projected center and depth, Tab. 1 of [166] shows that replacing ground truth depth with the predicted depth leads to inferior detection compared to replacing ground truth projected center with the predicted projected center. Hence, we conclude that depth is the hardest parameter to estimate.

C.2 Implementation Details

We now provide some additional implementation details for facilitating reproduction of this work.

C.2.1 Steerable Filters of SES Convolution

We use the scale equivariant steerable blocks proposed by [226] for our DEVIANT backbone. We now share the implementation details of these steerable filters.

Basis. Although steerable filters can use any linearly independent functions as their basis, we stick

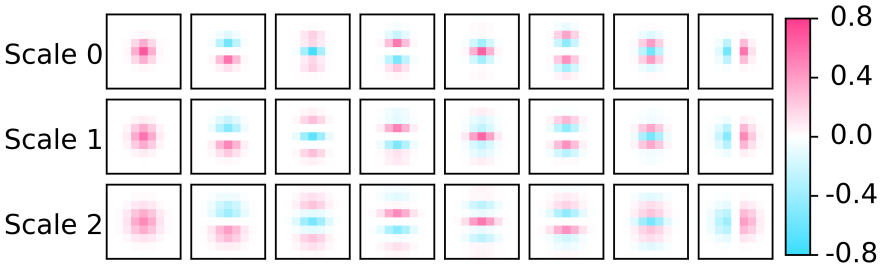


Figure C.6 **Steerable Basis** [227] for 7×7 SES convolution filters. (Showing only 8 of the 49 members for each scale).

with the Hermite polynomials as the basis [226]. Let $(0, 0)$ denote the center of the function and (u, v) denote the pixel coordinates. Then, the filter coefficients $\psi_{\sigma nm}$ [226] are

$$\psi_{\sigma nm} = \frac{A}{\sigma^2} H_n\left(\frac{u}{\sigma}\right) H_m\left(\frac{v}{\sigma}\right) e^{-\frac{u^2+v^2}{\sigma^2}} \tag{C.14}$$

H_n denotes the Probabilist's Hermite polynomial of the nth order, and A is the normalization constant. The first six Probabilist's Hermite polynomials are

$$H_0(x) = 1 \tag{C.15}$$

$$H_1(x) = x \tag{C.16}$$

$$H_2(x) = x^2 - 1 \tag{C.17}$$

$$H_3(x) = x^3 - 3x \tag{C.18}$$

$$H_4(x) = x^4 - 6x^2 + 3 \tag{C.19}$$

Fig. C.6 visualizes some of the SES filters and shows that the basis is indeed at different scales.

C.2.2 Monocular 3D Detection

Architecture. We use the DLA-34 [292] configuration, with the standard Feature Pyramid Network (FPN) [138], binning and ensemble of uncertainties. FPN is a bottom-up feed-forward CNN that computes feature maps with a downscaling factor of 2, and a top-down network that brings them back to the high-resolution ones. There are total six feature maps levels in this FPN.

We use DLA-34 as the backbone for our baseline GUP Net [159], while we use SES-DLA-34 as the backbone for DEVIANT. We also replace the 2D pools by 3D pools with pool along the scale dimensions as 1 for DEVIANT.

We initialize the vanilla CNN from ImageNet weights. For DEVIANT, we use the regularized least squares [226] to initialize the trainable weights in all the Hermite scales from the ImageNet [48] weights. Compared to initializing one of the scales as proposed in [226], we observed more stable convergence in initializing all the Hermite scales.

We output three foreground classes for KITTI dataset. We also output three foreground classes for Waymo dataset ignoring the Sign class [199].

Datasets. We use the publicly available KITTI,Waymo and nuScenes datasets for our experiments. KITTI is available at http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d under CC BY-NC-SA 3.0 License. Waymo is available at https://waymo.com/intl/en_us/dataset-download-terms/ under the Apache License, Version 2.0. nuScenes is available at https://www.nuscenes.org/nuscenes under CC BY-NC-SA 4.0 International Public License.

Augmentation. Unless otherwise stated, we horizontal flip the training images with probability 0.5, and use scale augmentation as 0.4 as well for all the models [159] in training.

Pre-processing. The only pre-processing step we use is image resizing.

- *KITTI.* We resize the $[370, 1242]$ sized KITTI images, and bring them to the $[384, 1280]$ resolution [159].
- *Waymo.* We resize the $[1280, 1920]$ sized Waymo images, and bring them to the $[512, 768]$ resolution. This resolution preserves their aspect ratio.

Box Filtering. We apply simple hand-crafted rules for filtering out the boxes. We ignore the box if it belongs to a class different from the detection class.

- *KITTI.* We train with boxes which are atleast $2m$ distant from the ego camera, and with visibility > 0.5 [159].
- *Waymo.* We train with boxes which are atleast $2m$ distant from the ego camera. The Waymo dataset does not have any occlusion based labels. However, Waymo provides the number of LiDAR points inside each 3D box which serves as a proxy for the occlusion. We train the boxes which have more than 100 LiDAR points for the vehicle class and have more than 50 LiDAR points for the cyclist and pedestrian class.

Training. We use the training protocol of GUP Net [159] for all our experiments. Training uses the Adam optimizer [102] and weight-decay 1×10^{-5} . Training dynamically weighs the losses using Hierarchical Task Learning (HTL) [159] strategy keeping K as 5 [159]. Training also uses a linear warmup strategy in the first 5 epochs to stabilize the training. We choose the model saved in the last epoch as our final model for all our experiments.

- *KITTI.* We train with a batch size of 12 on single Nvidia A100 (40GB) GPU for 140 epochs. Training starts with a learning rate 1.25×10^{-3} with a step decay of 0.1 at the 90th and the 120th epoch.
- *Waymo.* We train with a batch size of 40 on single Nvidia A100 (40GB) GPU for 30 epochs because of the large size of the Waymo dataset. Training starts with a learning rate 1.25×10^{-3} with a step decay of 0.1 at the 18th and the 26th epoch.

Losses. We use the GUP Net [159] multi-task losses before the NMS for training. The total loss $\mathcal{L}$ is given by

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{heatmap}} + \mathcal{L}_{\text{2D,offset}} + \mathcal{L}_{\text{2D,size}} + \mathcal{L}_{\text{3D2D,offset}} + \mathcal{L}_{\text{3D},angle} \\ + \mathcal{L}_{\text{3D},l} + \mathcal{L}_{\text{3D},w} + \mathcal{L}_{\text{3D},h} + \mathcal{L}_{\text{3D},depth}.\end{aligned} \tag{C.20}$$

The individual terms are given by

$$\mathcal{L}_{\text{heatmap}} = \text{Focal}(class^b, class^g), \tag{C.21}$$

$$\mathcal{L}_{\text{2D,offset}} = \mathcal{L}_1(\delta^b_{\text{2D}}, \delta^g_{\text{2D}}), \tag{C.22}$$

$$\mathcal{L}_{\text{2D,size}} = \mathcal{L}_1(w^b_{\text{2D}}, w^g_{\text{2D}}) + \mathcal{L}_1(h^b_{\text{2D}}, h^g_{\text{2D}}), \tag{C.23}$$

$$\mathcal{L}_{\text{3D2D,offset}} = \mathcal{L}_1(\delta^b_{\text{3D2D}}, \delta^g_{\text{3D2D}}) \tag{C.24}$$

$$\mathcal{L}_{\text{3D},angle} = \text{CE}(\alpha^b, \alpha^g) \tag{C.25}$$

$$\mathcal{L}_{\text{3D},l} = \mathcal{L}_1(\mu^b_{l\text{3D}}, \delta^g_{l\text{3D}}) \tag{C.26}$$

$$\mathcal{L}_{\text{3D},w} = \mathcal{L}_1(\mu^b_{w\text{3D}}, \delta^g_{w\text{3D}}) \tag{C.27}$$

$$\mathcal{L}_{\text{3D},h} = \frac{\sqrt{2}}{\sigma_{h\text{3D}}}\mathcal{L}_1(\mu^b_{h\text{3D}}, \delta^g_{h\text{3D}}) + \ln(\sigma_{h\text{3D}}) \tag{C.28}$$

$$\mathcal{L}_{\text{3D},depth} = \frac{\sqrt{2}}{\sigma_d}\mathcal{L}_1(\mu^b_d, \mu^g_d) + \ln(\sigma_d), \tag{C.29}$$

where,

$$\mu_d^b = f\frac{\mu_{h3\mathrm{D}}^b}{h_{2\mathrm{D}}^b} + \mu_{d,pred} \tag{C.30}$$

$$\sigma_d = \sqrt{\left(f\frac{\sigma_{h3\mathrm{D}}}{h_{2\mathrm{D}}^b}\right)^2 + \sigma_{d,pred}^2}. \tag{C.31}$$

The superscripts b and g denote the predicted box and ground truth box respectively. CE and Focal denote the Cross Entropy and Focal loss respectively.

The number of heatmaps depends on the number of output classes. $\delta_{2\mathrm{D}}$ denotes the deviation of the 2D center from the center of the heatmap. $\delta_{3\mathrm{D2D,offset}}$ denotes the deviation of the projected 3D center from the center of the heatmap. The orientation loss is the cross entropy loss between the binned observation angle of the prediction and the ground truth. The observation angle α is split into 12 bins covering 30° range. $\delta_{l3\mathrm{D}}, \delta_{w3\mathrm{D}}$ and $\delta_{h3\mathrm{D}}$ denote the deviation of the 3D length, width and height of the box from the class dependent mean size respectively.

The depth is the hardest parameter to estimate [166]. So, GUP Net uses in-network ensembles to predict the depth. It obtains a Laplacian estimate of depth from the 2D height, while it obtains another estimate of depth from the prediction of depth. It then adds these two depth estimates.

Inference. Our testing resolution is same as the training resolution. We do not use any augmentation for test/validation. We keep the maximum number of objects to 50 in an image, and we multiply the class and predicted confidence to get the box's overall score in inference as in [109]. We consider output boxes with scores greater than a threshold of 0.2 for KITTI [159] and 0.1 for Waymo [199].

C.3 Additional Experiments and Results

We now provide additional details and results of the experiments evaluating DEVIANT's performance.

C.3.1 KITTI Val Split

Monocular Detection has Huge Generalization Gap. As mentioned in Sec. 3.1, we now show that the monocular detection has huge generalization gap between training and inference. We report the object detection performance on the train and validation (val) set for the two models on KITTI Val split in Tab. C.2. Tab. C.2 shows that the performance of our baseline GUP Net [159] and our

Table C.2 **Generalization gap** (↓) on KITTI Val cars. Monocular detection has huge generalization gap between training and inference sets. [Key: **Best**]

Method	Scale Eqv	Set	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)		
			Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [159]		Train	91.83	74.87	67.43	95.19	80.95	73.55	99.50	93.62	86.22	99.56	93.88	86.46
		Val	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
		Gap	70.73	**59.39**	54.55	66.61	60.03	55.72	40.55	49.63	**48.15**	34.96	46.12	**43.49**
DEVIANT	✓	Train	91.09	76.19	67.16	94.76	82.61	75.51	99.37	93.56	88.57	99.50	93.87	88.90
		Val	24.63	16.54	14.52	32.60	23.04	19.99	61.00	46.00	40.18	65.28	49.63	43.50
		Gap	**66.46**	59.65	**52.64**	**62.16**	**59.57**	**55.52**	**38.37**	**47.56**	48.39	**34.22**	**44.24**	45.40

Table C.3 **Comparison on multiple backbones** on KITTI Val cars. [Key: **Best**]

BackBone	Method	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
		$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
ResNet-18	GUP Net [159]	18.86	13.20	11.01	26.05	19.37	16.57	54.90	40.65	34.98	60.54	46.13	40.12
	DEVIANT	20.27	14.21	12.56	28.09	20.32	17.49	55.75	42.41	36.97	60.82	46.43	40.59
DLA-34	GUP Net [159]	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
	DEVIANT	**24.63**	**16.54**	**14.52**	**32.60**	**23.04**	**19.99**	**61.00**	**46.00**	**40.18**	**65.28**	**49.63**	**43.50**

DEVIANT is huge on the training set, while it is less than one-fourth of the train performance on the val set.

We also report the generalization gap (in pink) metric [270] in Tab. C.2, which is the difference between training and validation performance. The generalization gap at both the thresholds of 0.7 and 0.5 is huge.

Comparison on Multiple Backbones. A common trend in 2D object detection community is to show improvements on multiple backbones [253]. DD3D [181] follows this trend and also reports their numbers on multiple backbones. Therefore, we follow the same and compare with our baseline on multiple backbones on KITTI Val cars in Tab. C.3. Tab. C.3 shows that DEVIANT shows consistent improvements over GUP Net [159] in 3D object detection on multiple backbones, proving the effectiveness of our proposal.

Comparison with Bigger CNN Backbones. Since the SES blocks increase the Flop counts significantly compared to the vanilla convolution block, we next compare DEVIANT with bigger CNN backbones with comparable GFLOPs and FPS/ wall-clock time (instead of same configuration) in Tab. C.4. We compare DEVIANT with DLA-102 and DLA-169 - two biggest DLA networks

Table C.4 **Results with bigger CNNs having similar flops** on KITTI Val cars. [Key: **Best**]

Method	BackBone	Param (↓) (M)	Disk Size (↓) (MB)	Flops (↓) (G)	Infer (↓) (ms)	AP_{3D} $IoU_{3D} \geq 0.7$ (↑) Easy	Mod	Hard	AP_{3D} $IoU_{3D} \geq 0.5$ (↑) Easy	Mod	Hard
GUP Net [159]	DLA-34	**16**	**235**	**30**	**20**	21.10	15.48	12.88	58.95	43.99	38.07
GUP Net [159]	DLA-102	34	583	70	25	20.96	14.64	12.80	57.06	41.78	37.26
GUP Net [159]	DLA-169	54	814	114	30	21.76	15.35	12.72	57.60	43.27	37.32
DEVIANT	SES-DLA-34	**16**	236	235	40	**24.63**	**16.54**	**14.52**	**61.00**	**46.00**	**40.18**

Table C.5 **Results on KITTI Val cyclists and pedestrians** (Cyc/Ped) ($IoU_{3D} \geq 0.5$). [Key: **Best**, **Second Best**]

Method	Extra	Cyc AP $_{3D\|R_{40}}$ [%](↑) Easy	Mod	Hard	Ped AP $_{3D\|R_{40}}$ [%](↑) Easy	Mod	Hard
GrooMeD-NMS [109]	–	0.00	0.00	0.00	3.79	2.71	2.61
MonoDIS [222]	–	1.52	0.73	0.71	3.20	2.28	1.71
MonoDIS-M [220]	–	2.70	1.50	1.30	9.50	7.10	5.70
GUP Net (Retrained) [159]	–	**4.41**	**2.17**	**2.03**	**9.37**	**6.84**	**5.73**
DEVIANT (Ours)	–	**4.05**	**2.20**	**2.14**	**9.85**	**7.18**	**5.42**

with ImageNet weights[1] on KITTI Val split. We use the fvcore library[2] to get the parameters and flops. Tab. C.4 shows that DEVIANT again outperforms the bigger CNN backbones, especially on nearby objects. We believe this happens because the bigger CNN backbones have more trainable parameters than DEVIANT, which leads to overfitting. Although DEVIANT takes more time compared to the CNN backbones, DEVIANT still keeps the inference almost real-time.

Performance on Cyclists and Pedestrians. Tab. C.5 lists out the results of 3D object detection on KITTI Val Cyclist and Pedestrians. The results show that DEVIANT is competitive on challenging Cyclist and achieves SoTA results on Pedestrians on the KITTI Val split.

Cross-Dataset Evaluation Details. For cross-dataset evaluation, we test on all 3,769 images of the KITTI Val split, as well as all frontal 6,019 images of the nuScenes Val split [22], as in [218]. We first convert the nuScenes Val images to the KITTI format using the `export_kitti`[3] function in the nuscenes devkit. We keep KITTI Val images in the $[384, 1280]$ resolution, while we keep the nuScenes Val images in the $[384, 672]$ resolution to preserve the aspect ratio. For M3D-RPN [15], we bring the nuScenes Val images in the $[512, 910]$ resolution.

Monocular 3D object detection relies on the camera focal length to back-project the projected

[1] Available at http://dl.yf.io/dla/models/imagenet/
[2] https://github.com/facebookresearch/fvcore
[3] https://github.com/nutonomy/nuscenes-devkit/blob/master/python-sdk/nuscenes/scripts/export_kitti.py

centers into the 3D space. Therefore, the 3D centers depends on the focal length of the camera used in the dataset. Hence, one should take the camera focal length into account while doing cross-dataset evaluation. We now calculate the camera focal length of a dataset as follows. We take the camera matrix $\mathbf{K}$ and calculate the normalized focal length $\bar{f} = \frac{2f_y}{H}$, where H denotes the height of the image. The normalized focal length $\bar{f}$ for the KITTI dataset is 3.82, while the normalized focal length $\bar{f}$ for the nuScenes dataset is 2.82. Thus, the KITTI and the nuScenes images have a different focal length [255].

M3D-RPN [15] does not normalize w.r.t. the focal length. So, we explicitly correct and divide the depth predictions of nuScenes images from the KITTI model by $3.82/2.82 = 1.361$ in the M3D-RPN [15] codebase. The GUP Net [159] and DEVIANT codebases use normalized coordinates *i.e.* they normalize w.r.t. the focal length. So, we do not explicitly correct the focal length for GUP Net and DEVIANT predictions.

We match predictions to the ground truths using the IoU_{2D} overlap threshold of 0.7 [218]. After this matching, we calculate the Mean Average Error (MAE) of the depths of the predicted and the ground truth boxes [218].

Stress Test with Rotational and/or xy-translation Ego Movement. Corollary 1.1 uses translation along the depth as the sole ego movement. This assumption might be valid for the current outdoor datasets and benchmarks, but is not the case in the real world. Therefore, we conduct stress tests on how tolerable DEVIANT and GUP Net [159] are when there is rotational and/or xy-translation movement on the vehicle.

First, note that KITTI and Waymo are already large-scale real-world datasets, and our own dataset might not be a good choice. So, we stick with KITTI and Waymo datasets. We manually choose 306 KITTI Val images with such ego movements and again compare performance of DEVIANT and GUP Net on this subset in Tab. C.6. The average distance of the car in this subset is 27.69 m (±16.59 m), which suggests a good variance and unbiasedness in the subset. Tab. C.6 shows that both the DEVIANT backbone and the CNN backbone show a drop in the detection performance by about 4 AP points on the Mod cars of ego-rotated subset compared to the all set.

Table C.6 **Stress Test** with rotational and xy-translation ego movement on KITTI Val cars. [Key: **Best**]

Set	Method	AP_{3D} $IoU_{3D} \geq 0.7$ (↑) Easy	Mod	Hard	AP_{3D} $IoU_{3D} \geq 0.5$ (↑) Easy	Mod	Hard
Subset (306)	GUP Net [159]	17.22	11.43	9.91	47.47	35.02	32.63
	DEVIANT	**20.17**	**12.49**	**10.93**	**49.81**	**36.93**	**34.32**
KITTI Val (3769)	GUP Net [159]	21.10	15.48	12.88	58.95	43.99	38.07
	DEVIANT	**24.63**	**16.54**	**14.52**	**61.00**	**46.00**	**40.18**

Table C.7 **Comparison of Depth Estimates** of monocular depth estimators and 3D object detectors on KITTI Val cars. Depth from a depth estimator BTS is not good for foreground objects (cars) beyond 20+ m range. [Key: **Best**, **Second Best**]

Method	Depth at	Ground Truth	Back+ Foreground 0−20	20−40	40−∞	Foreground (Cars) 0−20	20−40	40−∞
GUP Net [159]	3D Center	3D Box	–	–	–	0.45	**1.10**	**1.85**
DEVIANT	3D Center	3D Box	–	–	–	**0.40**	**1.09**	**1.80**
BTS [118]	Pixel	LiDAR	0.48	1.30	1.83	**0.30**	1.22	2.16

This drop experimentally confirms the theory that both the DEVIANT backbone and the CNN backbone do not handle arbitrary 3D rotations. More importantly, the table shows that DEVIANT maintains the performance improvement over GUP Net [159] under such movements.

Also, Waymo has many images in which the ego camera shakes. Improvements on Waymo (Tab. 3.12) also confirms that DEVIANT outperforms GUP Net [159] even when there is rotational or xy-translation ego movement.

Comparison of Depth Estimates from Monocular Depth Estimators and 3D Object Detectors. We next compare the depth estimates from monocular depth estimators and depth estimates from monocular 3D object detectors on the foreground objects. We take a monocular depth estimator BTS [118] model trained on KITTI Eigen split. We next compare the depth error for all and foreground objects (cars) on KITTI Val split using MAE (↓) metric in Tab. C.7 as in Tab. 3.6. We use the MSeg [114] to segment out cars in the driving scenes for BTS. Tab. C.7 shows that the depth from BTS is not good for foreground objects (cars) beyond 20+ m range. Note that there is a data leakage issue between the KITTI Eigen train split and the KITTI Val split [221] and therefore, we expect more degradation in performance of monocular depth estimators after fixing the data leakage issue.

Equivariance Error for KITTI Monocular Videos. A better way to compare the scale equiv-

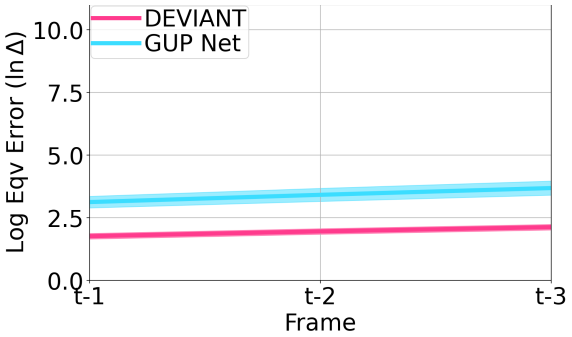


Figure C.7 **Equivariance error** (Δ) comparison for DEVIANT and GUP Net on previous three frames of the KITTI monocular videos at block 3 in the backbone.

ariance of the DEVIANT and GUP Net [159] compared to Fig. 3.4, is to compare equivariance error on real images with depth translations of the ego camera. The equivariance error Δ is the normalized difference between the scaled feature map and the feature map of the scaled image, and is given by

$$\Delta = \frac{1}{N} \sum_{i=1}^{N} \frac{||\mathcal{T}_{s_i} \Phi(h_i) - \Phi(\mathcal{T}_{s_i} h_i)||_2^2}{||\mathcal{T}_{s_i} \Phi(h_i)||_2^2}, \tag{C.32}$$

where Φ denotes the neural network, $\mathcal{T}_{s_i}$ is the scaling transformation for the image i, and N is the total number of images. Although we do evaluate this error in Fig. 3.4, the image scaling in Fig. 3.4 does not involve scene change because of the absence of the moving objects. Therefore, evaluating on actual depth translations of the ego camera makes the equivariance error evaluation more realistic. We next carry out this experiment and report the equivariance error on three previous frames of the val images of the KITTI Val split as in [17]. We plot this equivariance error in Fig. C.7 at block 3 of the backbones because the resolution at this block corresponds to the output feature map of size $[96, 320]$. Fig. C.7 is similar to Fig. 3.4b, and shows that DEVIANT achieves lower equivariance error. Therefore, DEVIANT has better equivariance to depth translations (scale transformation s) than GUP Net [159] in real scenarios.

Model Size, Training, and Inference Times. Both DEVIANT and the baseline GUP Net have the same number of trainable parameters, and therefore, the same model size. GUP Net takes 4 hours to train on KITTI Val and 0.02 ms per image for inference on a single Ampere A100 (40 GB) GPU. DEVIANT takes 8.5 hours for training and 0.04 ms per image for inference on the same GPU. This

Table C.8 **Five Different Runs** on KITTI Val cars. [Key: **Average**]

Method	Run	$IoU_{3D} \geq 0.7$						$IoU_{3D} \geq 0.5$					
		$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)			$AP_{3D\|R_{40}}$ [%](↑)			$AP_{BEV\|R_{40}}$ [%](↑)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [159]	1	21.67	14.75	12.68	28.72	20.88	17.79	58.27	43.53	37.62	63.67	47.37	42.55
	2	21.26	14.94	12.49	28.39	20.40	17.43	59.20	43.55	37.63	64.06	47.46	42.67
	3	20.87	15.03	12.61	28.66	20.56	17.48	60.19	44.08	39.36	65.26	49.44	43.17
	4	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
	5	22.52	15.92	13.31	30.77	22.40	19.36	59.91	44.00	39.30	64.94	48.01	43.08
	Avg	**21.48**	**15.22**	**12.79**	**29.02**	**21.03**	**17.98**	**59.30**	**43.83**	**38.40**	**64.51**	**48.01**	**42.89**
DEVIANT	1	23.19	15.84	14.11	29.82	21.93	19.16	60.19	45.52	39.86	66.32	49.39	43.38
	2	23.33	16.12	13.54	31.22	22.64	19.64	61.59	46.33	40.35	67.49	50.26	43.98
	3	24.12	16.37	14.48	31.58	22.52	19.65	62.51	46.47	40.65	67.33	50.24	44.16
	4	24.63	16.54	14.52	32.60	23.04	19.99	61.00	46.00	40.18	65.28	49.63	43.50
	5	25.82	17.69	15.07	33.63	23.84	20.60	62.39	46.46	40.61	67.55	50.51	45.80
	Avg	**24.22**	**16.51**	**14.34**	**31.77**	**22.79**	**19.81**	**61.54**	**46.16**	**40.33**	**66.79**	**50.01**	**44.16**

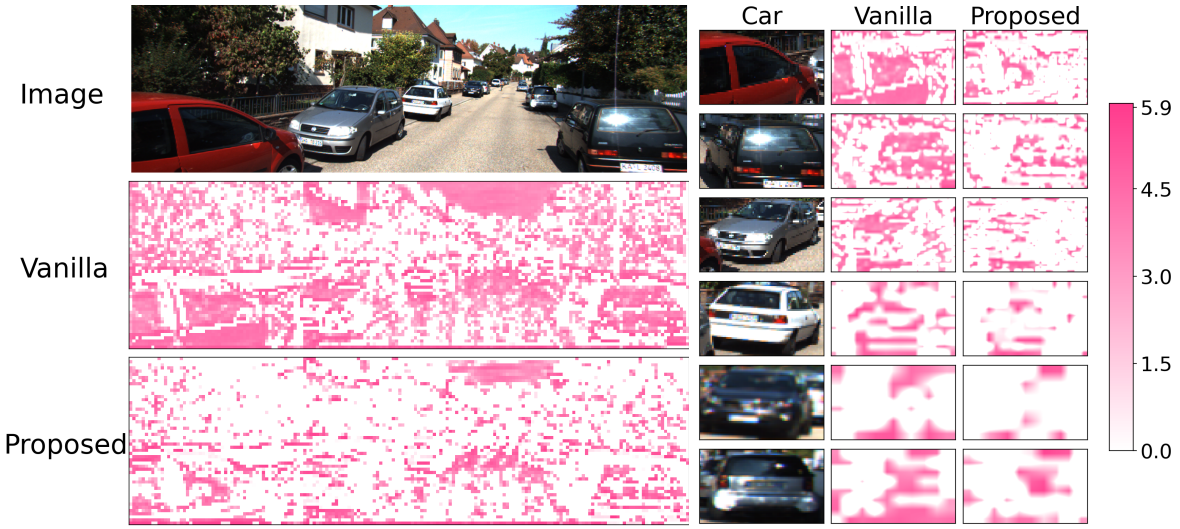
Table C.9 **Experiments Comparison**.

Method	Venue	Multi-Dataset	Cross-Dataset	Multi-Backbone
GrooMeD-NMS [109]	CVPR21	–	–	–
MonoFlex [301]	CVPR21	–	–	–
CaDDN [199]	CVPR21	✓	–	–
MonoRCNN [218]	ICCV21	–	✓	–
GUP Net [159]	ICCV21	–	–	–
DD3D [181]	ICCV21	✓	–	✓
PCT [246]	NeurIPS21	✓	–	✓
MonoDistill [39]	ICLR22	–	–	–
MonoDIS-M [220]	TPAMI20	✓	–	–
MonoEF [307]	TPAMI21	✓	–	–
DEVIANT	-	✓	✓	✓

is expected because SE models use more flops [227, 309] and, therefore, DEVIANT takes roughly twice the training and inference time as GUP Net.

Reproducibility. As described in Sec. 3.5.2, we now list out the five runs of our baseline GUP Net [159] and DEVIANT in Tab. C.8. Tab. C.8 shows that DEVIANT outperforms GUP Net in all runs and in the average run.

Experiment Comparison. We now compare the experiments of different chapters in Tab. C.9. To the best of our knowledge, the experimentation in DEVIANT is more than the experimentation of most monocular 3D object detection chapters.



(a) Depth equivariance error (↓). (b) Error (↓) on objects.

Figure C.8 **(a) Depth (scale) equivariance error** of vanilla GUP Net [159] and proposed DEVIANT. (See Sec. 3.5.2 for details) **(b) Error on objects.** The proposed backbone has less depth equivariance error than vanilla CNN backbone.

C.3.2 Qualitative Results

KITTI. We next show some more qualitative results of models trained on KITTI Val split in Fig. C.9. We depict the predictions of DEVIANT in image view on the left and the predictions of DEVIANT and GUP Net [159], and ground truth in BEV on the right. In general, DEVIANT predictions are more closer to the ground truth than GUP Net [159].

nuScenes Cross-Dataset Evaluation. We then show some qualitative results of KITTI Val model evaluated on nuScenes frontal in Fig. C.10. We again observe that DEVIANT predictions are more closer to the ground truth than GUP Net [159]. Also, considerably less number of boxes are detected in the cross-dataset evaluation *i.e.* on nuScenes. We believe this happens because of the domain shift.

Waymo. We now show some qualitative results of models trained on Waymo Val split in Fig. C.11. We again observe that DEVIANT predictions are more closer to the ground truth than GUP Net [159].

C.3.3 Demo Videos of DEVIANT

Detection Demo. We next put a short demo video of our DEVIANT model trained on KITTI Val split at https://www.youtube.com/watch?v=2D73ZBrU-PA. We run our trained model indepen-

dently on each frame of 2011_09_26_DRIVE_0009 KITTI raw [66]. The video belongs to the City category of the KITTI raw video. None of the frames from the raw video appear in the training set of KITTI Val split [109]. We use the camera matrices available with the video but do not use any temporal information. Overlaid on each frame of the raw input videos, we plot the projected 3D boxes of the predictions and also plot these 3D boxes in the BEV. We set the frame rate of this demo at 10 fps as in KITTI. The attached demo video demonstrates very stable and impressive results because of the additional equivariance to depth translations in DEVIANT which is absent in vanilla CNNs. Also, notice that the orientation of the boxes are stable despite not using any temporal information.

Equivariance Error Demo. We next show the depth equivariance (scale equivariance) error demo of one of the channels from the vanilla GUP Net and our proposed method at https://www.youtube.com/watch?v=70DIjQkuZvw. As before, we report at block 3 of the backbones which corresponds to output feature map of the size $[96, 320]$. The equivariance error demo indicates more white spaces which confirms that DEVIANT achieves lower equivariance error compared to the baseline GUP Net [159]. Thus, this demo agrees with Fig. C.8a. This happens because depth (scale) equivariance is additionally hard-baked into DEVIANT, while the vanilla GUP Net is not equivariant to depth translations (scale transformation s).

Figure C.9 **KITTI Qualitative Results**. DEVIANT predictions in general are more accurate than GUP Net [159]. [Key: Cars (pink), Cyclists (orange) and Pedestrians (violet) of DEVIANT; all classes of GUP Net (cyan), and Ground Truth (green) in BEV].

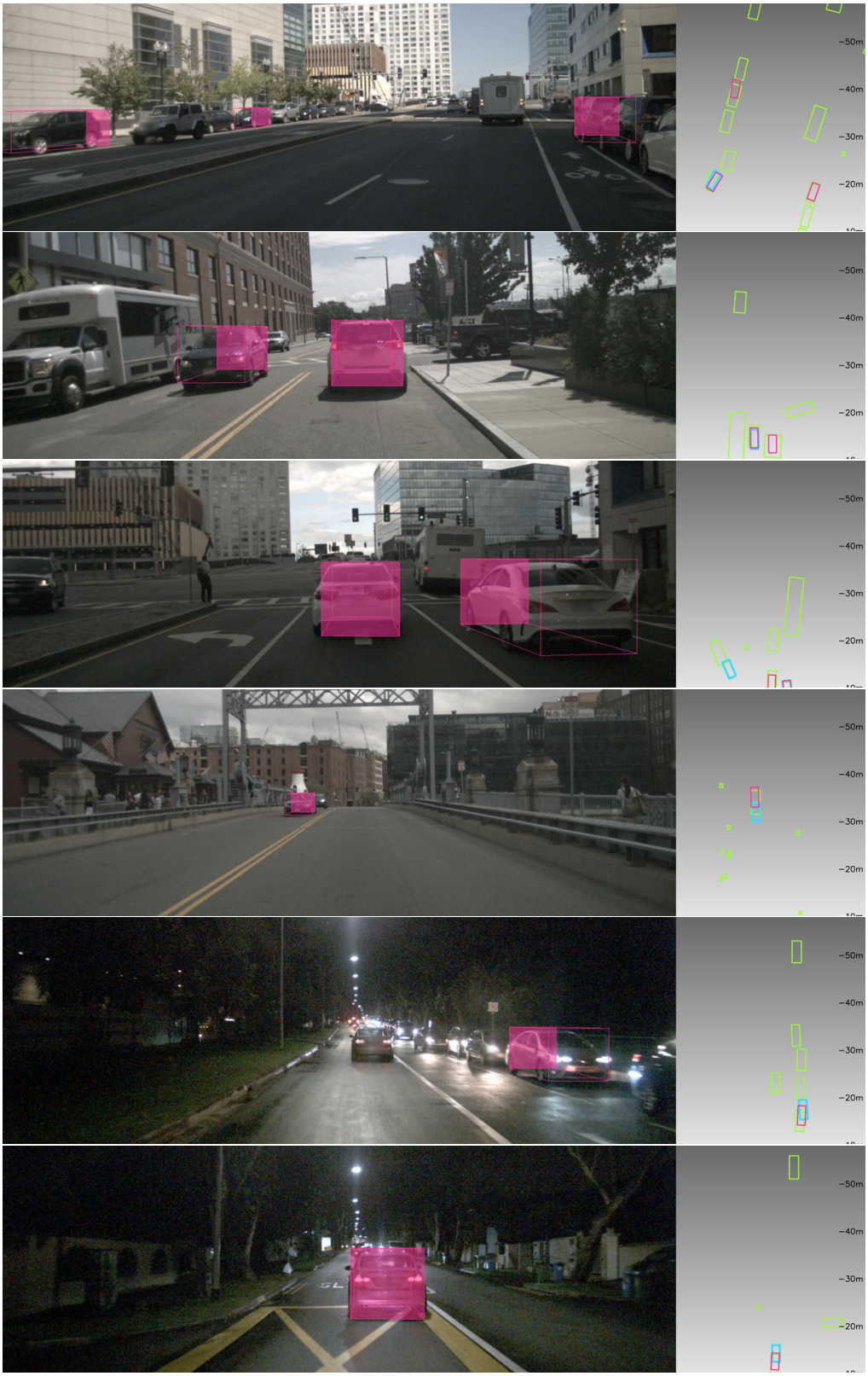


Figure C.10 **nuScenes Cross-Dataset Qualitative Results.** DEVIANT predictions in general are more accurate than GUP Net [159]. [Key: Cars of DEVIANT (pink); Cars of GUP Net (cyan), and Ground Truth (green) in BEV].

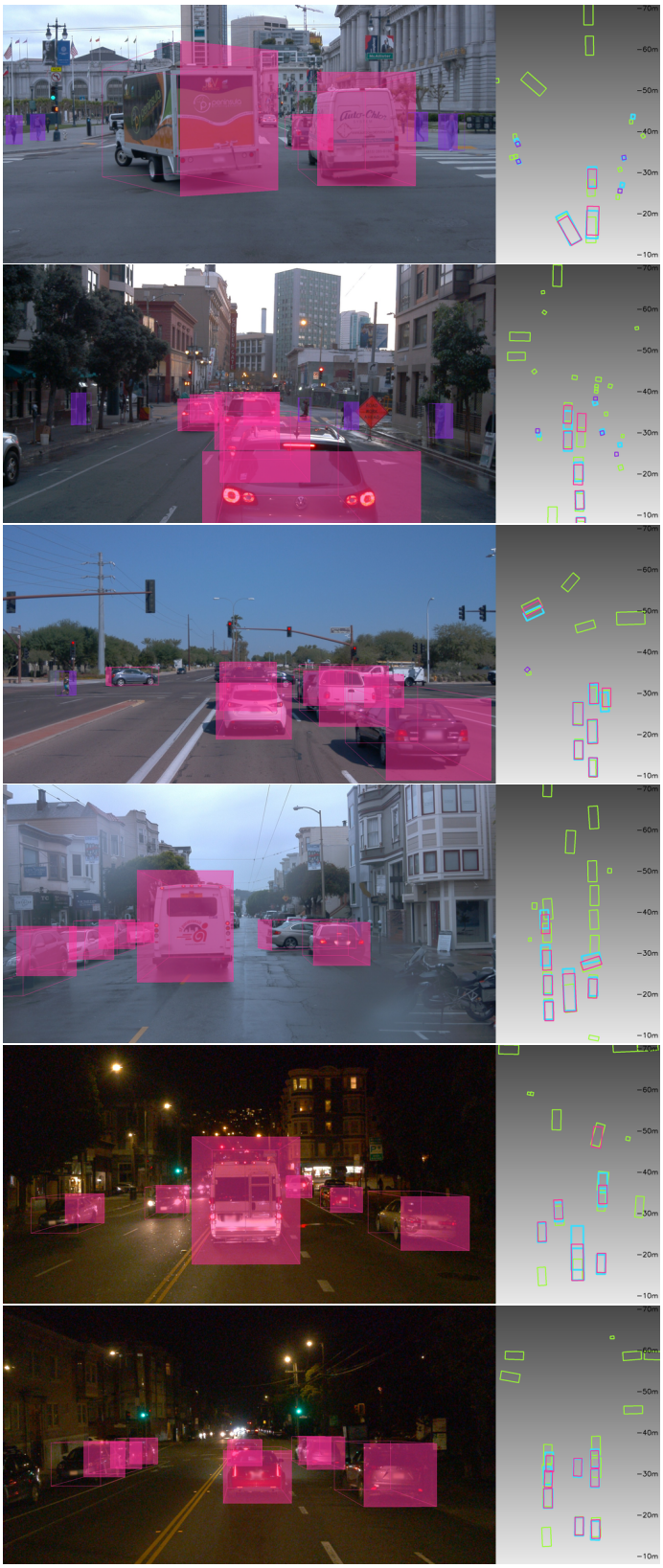


Figure C.11 **Waymo Qualitative Results**. DEVIANT predictions in general are more accurate than GUP Net [159]. [Key: Cars (pink), Cyclists (orange) and Pedestrians (violet) of DEVIANT; all classes of GUP Net (cyan), and Ground Truth (green) in BEV].

APPENDIX D

SEABIRD APPENDIX

D.1 Additional Explanations and Proofs

We now add some explanations and proofs which we could not put in the main chapter because of the space constraints.

D.1.1 Proof of Converged Value

We first bound the converged value from the optimal value. These results are well-known in the literature [113, 214]. We reproduce the result from using our notations for completeness.

$$\begin{aligned}
&\mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \\
&= \mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu} + {}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*\right\|_2^2\right) \\
&= \mathbb{E}\left(\left({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu} + {}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*\right)^T \left({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu} + {}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*\right)\right) \\
&= \mathbb{E}(({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu})^T({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu})) + \mathbb{E}(({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)^T({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)) \\
&\quad + 2\mathbb{E}(({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu})^T({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)) \\
&= \mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) + \mathbb{E}(({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)^T({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*))
\end{aligned} \tag{D.1}$$

where ${}^{\mathcal{L}}\boldsymbol{\mu} = \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_\infty)$ is the mean of the layer weight and $\mathrm{Var}(\mathbf{w})$ denotes the variance of $\sum_j w_j^2$.

SGD. We begin the proof by writing the value of ${}^{\mathcal{L}}\mathbf{w}_t$ at every step. The model uses SGD, and so, the weight ${}^{\mathcal{L}}\mathbf{w}_t$ after t gradient updates is

$${}^{\mathcal{L}}\mathbf{w}_t = \mathbf{w}_0 - s_1{}^{\mathcal{L}}\mathbf{g}_1 - s_2{}^{\mathcal{L}}\mathbf{g}_2 - \cdots - s_t{}^{\mathcal{L}}\mathbf{g}_t, \tag{D.2}$$

where ${}^{\mathcal{L}}\mathbf{g}_t$ denotes the gradient of $\mathbf{w}$ at every step t. Assume the loss function under consideration $\mathcal{L}$ is $\mathcal{L} = f(\mathbf{w}_t\mathbf{h} - z) = f(\eta)$. Then, we have,

$$\begin{aligned}
{}^{\mathcal{L}}\mathbf{g}_t &= \frac{\partial \mathcal{L}}{\partial \mathbf{w}_t} \\
&= \frac{\partial \mathcal{L}(\mathbf{w}_t\mathbf{h} - z)}{\partial \mathbf{w}_t} \\
&= \frac{\partial \mathcal{L}(\mathbf{w}_t\mathbf{h} - z)}{\partial (\mathbf{w}_t\mathbf{h} - z)} \frac{\partial (\mathbf{w}_t\mathbf{h} - z)}{\partial \mathbf{w}_t}
\end{aligned}$$

$$= \frac{\partial \mathcal{L}(\eta)}{\partial \eta}\mathbf{h}$$
$$= \mathbf{h}\frac{\partial \mathcal{L}(\eta)}{\partial \eta}$$
$$\implies {}^{\mathcal{L}}\mathbf{g}_t = \mathbf{h}\epsilon, \tag{D.3}$$

with $\epsilon = \dfrac{\partial \mathcal{L}(\eta)}{\partial \eta}$ is the gradient of the loss function wrt noise.

Expectation and Variance of Gradient ${}^{\mathcal{L}}\mathbf{g}_t$ Since the image $\mathbf{h}$ and noise η are statistically independent, the image and the noise gradient η are also statistically independent. So, the expected gradients

$$\mathbb{E}({}^{\mathcal{L}}\mathbf{g}_t) = \mathbb{E}(\mathbf{h})\mathbb{E}(\epsilon) = 0. \tag{D.4}$$

Note that if the loss function is an even function (symmetric about zero), its gradient ϵ is an odd function (anti-symmetric about 0), and so its mean $\mathbb{E}(\epsilon) = 0$.

Next, we write the gradient variance $\mathrm{Var}({}^{\mathcal{L}}\mathbf{g}_t)$ as

$$\begin{aligned}\mathrm{Var}({}^{\mathcal{L}}\mathbf{g}_t) = \mathrm{Var}(\mathbf{h}\epsilon) &= \mathbb{E}(\mathbf{h}^T\mathbf{h})\mathbb{E}(\epsilon^2) - \mathbb{E}^2(\mathbf{h})\mathbb{E}^2(\epsilon) \\ &= \mathbb{E}(\mathbf{h}^T\mathbf{h})\left[\mathrm{Var}(\epsilon) + \mathbb{E}^2(\epsilon)\right] \\ &\quad - \mathbb{E}^2(\mathbf{h})\mathbb{E}^2(\epsilon) \\ \implies \mathrm{Var}({}^{\mathcal{L}}\mathbf{g}_t) &= \mathbb{E}(\mathbf{h}^T\mathbf{h})\mathrm{Var}(\epsilon) \quad \text{as } \mathbb{E}(\epsilon) = 0\end{aligned} \tag{D.5}$$

Expectation and Variance of Converged Weight ${}^{\mathcal{L}}\mathbf{w}_t$ We first calculate the expected converged weight as

$$\begin{aligned}\mathbb{E}({}^{\mathcal{L}}\mathbf{w}_t) &= \mathbb{E}(\mathbf{w}_0) + \left(\sum_{j=1}^{t} s_j \mathbb{E}\left({}^{\mathcal{L}}\mathbf{g}_j\right)\right), \text{using Eq. (D.2)} \\ &= \mathbf{0} \quad \text{using Eq. (D.4)} \\ \implies \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_\infty) &= \lim_{t\to\infty} \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_t) \\ \implies \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_\infty) &= {}^{\mathcal{L}}\boldsymbol{\mu} = \mathbf{0}\end{aligned} \tag{D.6}$$

We finally calculate the variance of the converged weight. Because the SGD step size is independent of the gradient, we write using Eq. (D.2),

$$\begin{aligned}\mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_t) = \mathrm{Var}(\mathbf{w}_0) + s_1^2\mathrm{Var}\left(\mathbf{g}_1\right) + s_2^2\mathrm{Var}\left(\mathbf{g}_2\right) \\ + \cdots + s_t^2\mathrm{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right)\end{aligned} \tag{D.7}$$

Assuming the gradients ${}^{\mathcal{L}}\mathbf{g}_t$ are drawn from an identical distribution, we have

$$\begin{aligned}&\mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_t) = \mathrm{Var}(\mathbf{w}_0) + \left(\sum_{j=1}^{t} s_j^2\right)\mathrm{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right) \\ \implies &\mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) = \lim_{t\to\infty}\mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_t) \\ &\qquad = \mathrm{Var}(\mathbf{w}_0) + \left(\lim_{t\to\infty}\sum_{j=1}^{t} s_j^2\right)\mathrm{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right) \\ \implies &\mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) = \mathrm{Var}(\mathbf{w}_0) + s\mathrm{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right)\end{aligned} \tag{D.8}$$

An example of square summable step-sizes of SGD is $s_j = \frac{1}{j}$, and then the constant $s = \sum_{j=1} s_j^2 = \frac{\pi^2}{6}$. This assumption is also satisfied by modern neural networks since their training steps are always finite.

Substituting Eq. (D.5) in Eq. (D.8), we have

$$\mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) = \mathrm{Var}(\mathbf{w}_0) + s\mathbb{E}(\mathbf{h}^T\mathbf{h})\mathrm{Var}(\epsilon) \tag{D.9}$$

Substituting mean and variances from Eqs. (D.6) and (D.9) in Eq. (D.1), we have

$$\begin{aligned}\mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) &= \mathrm{Var}(\mathbf{w}_0) + s\mathbb{E}(\mathbf{h}^T\mathbf{h})\mathrm{Var}(\epsilon) \\ &\quad + \mathbb{E}(||\mathbf{w}_*||^2) \\ &= s\mathbb{E}(\mathbf{h}^T\mathbf{h})\mathrm{Var}(\epsilon) + \mathrm{Var}(\mathbf{w}_0) \\ &\quad + \mathbb{E}(||\mathbf{w}_*||^2) \\ \implies \mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) &= c_1\mathrm{Var}(\epsilon) + c_2,\end{aligned} \tag{D.10}$$

where $\epsilon = \dfrac{\partial\mathcal{L}(\eta)}{\partial\eta}$ is the gradient of the loss function wrt noise, and $c_1 = s\mathbb{E}(\mathbf{h}^T\mathbf{h})$ and c_2 are terms independent of the loss function $\mathcal{L}$.

D.1.2 Comparison of Loss Functions

Eq. (4.1) shows that different losses $\mathcal{L}$ lead to different $\text{Var}(\epsilon)$. Hence, comparing this term for different losses asseses the quality of losses.

D.1.2.1 Gradient Variance of MAE Loss

The result on MAE ($\mathcal{L}_1$) is well-known in the literature [113, 214]. We reproduce the result from [113, 214] using our notations for completeness.

The $\mathcal{L}_1$ loss is

$$\begin{aligned} &\mathcal{L}_1(\eta) = |\hat{z} - z|_1 = |^{\mathcal{L}}\mathbf{w}_t\mathbf{h} - z|_1 = |\eta|_1 \\ &\implies \epsilon = \frac{\partial \mathcal{L}_1(\eta)}{\partial \eta} = \text{sgn}(\eta) \end{aligned} \tag{D.11}$$

Thus, $\epsilon = \text{sgn}(\eta)$ is a Bernoulli random variable with $p(\epsilon) = 1/2$ for $\epsilon = \pm 1$. So, mean $\mathbb{E}(\epsilon) = 0$ and variance $\text{Var}(\epsilon) = 1$.

D.1.2.2 Gradient Variance of MSE Loss

The result on MSE ($\mathcal{L}_2$) is well-known in the literature [113, 214]. We reproduce the result from [113, 214] using our notations for completeness. The $\mathcal{L}_2$ loss is

$$\begin{aligned} &\mathcal{L}_2(\eta) = 0.5|\hat{z} - z|^2 = 0.5|\eta|^2 = 0.5\eta^2 \\ &\implies \epsilon = \frac{\partial \mathcal{L}_2(\eta)}{\partial \eta} = \eta \end{aligned} \tag{D.12}$$

Thus, $\epsilon = \eta$ is a normal random variable [214]. So, mean $\mathbb{E}(\epsilon) = 0$ and variance $\text{Var}(\epsilon) = \text{Var}(\eta) = \sigma^2$.

D.1.2.3 Gradient Variance of Dice Loss. (Proof of Lemma 2)

Proof. We first write the gradient of dice loss as a function of noise (η) as follows:

$$\epsilon = \frac{\partial \mathcal{L}_{dice}(\eta)}{\partial \eta} = \begin{cases} \dfrac{\text{sgn}(\eta)}{\ell} & , |\eta| \leq \ell \\ 0 & , |\eta| \geq \ell \end{cases} \tag{D.13}$$

The gradient of the loss ϵ is an odd function and so, its mean $\mathbb{E}(\epsilon) = 0$. Next, we write its variance $\text{Var}(\epsilon)$ as

$$\begin{aligned}
\text{Var}(\epsilon) = \text{Var}(\eta) &= \frac{1}{\ell^2} \int_{-\ell}^{\ell} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\eta^2}{2\sigma^2}} d\eta \\
&= \frac{2}{\ell^2} \int_{0}^{\ell} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\eta^2}{2\sigma^2}} d\eta \\
&= \frac{2}{\ell^2} \int_{0}^{\ell/\sigma} \frac{1}{\sqrt{2\pi}} e^{-\frac{\eta^2}{2}} d\eta \\
&= \frac{2}{\ell^2} \left[\int_{-\infty}^{\ell/\sigma} \frac{1}{\sqrt{2\pi}} e^{-\frac{\eta^2}{2}} d\eta - \frac{1}{2} \right] \\
&= \frac{2}{\ell^2} \left[\Phi\left(\frac{\ell}{\sigma}\right) - \frac{1}{2} \right] \qquad \text{(D.14)}
\end{aligned}$$

where, Φ is the normal CDF

We write the CDF $\Phi(x)$ in terms of error function Erf as:

$$\Phi(x) = \frac{1}{2} + \frac{1}{2}\text{Erf}\left(\frac{x}{\sqrt{2}}\right) \qquad \text{(D.15)}$$

for $x \geq 0$. Next, we put $x = \dfrac{\ell}{\sigma}$ to get

$$\Phi\left(\frac{\ell}{\sigma}\right) = \frac{1}{2} + \frac{1}{2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \qquad \text{(D.16)}$$

Substituting above in Eq. (D.14), we obtain

$$\begin{aligned}
\text{Var}(\epsilon) &= \frac{2}{\ell^2} \left[\frac{1}{2} + \frac{1}{2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) - \frac{1}{2} \right] \\
\implies \text{Var}(\epsilon) &= \frac{1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \qquad \text{(D.17)}
\end{aligned}$$

D.1.3 Proof of Dice Model Being BetterLemma 3

Proof. It remains sufficient to show that

$$\mathbb{E}\left(\left\|{}^{d}\mathbf{w}_{\infty} - \mathbf{w}_{*}\right\|_2\right) \leq \mathbb{E}\left(\|{}^{r}\mathbf{w}_{\infty} - \mathbf{w}_{*}\|_2\right)$$

$$\implies \mathbb{E}\left(\left\|^{d}\mathbf{w}_{\infty}-\mathbf{w}_{*}\right\|_{2}^{2}\right) \leq \mathbb{E}\left(\|^{r}\mathbf{w}_{\infty}-\mathbf{w}_{*}\|_{2}^{2}\right) \tag{D.18}$$

Using Lemma 1, the above comparison is a comparison between the gradient variance of the loss wrt noise $\mathrm{Var}(\epsilon)$. Hence, we compute the gradient variance of the loss $\mathcal{L}$, *i.e.*, $\mathrm{Var}(\epsilon)$ of regression and dice losses to derive this lemma.

Case 1 $\sigma \leq 1$**:** Given Tab. 4.1, if $\sigma \leq 1$, the minimum deviation in converged regression model comes from the $\mathcal{L}_2$ loss. The difference in the estimates of regression loss and the dice loss

$$\begin{aligned}\mathbb{E}\left(\|^{r}\mathbf{w}_{\infty}-\mathbf{w}_{*}\|_{2}^{2}\right)-\mathbb{E}\left(\left\|^{d}\mathbf{w}_{\infty}-\mathbf{w}_{*}\right\|_{2}^{2}\right)\\ \propto \sigma^{2}-\frac{1}{\ell^{2}}\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)\end{aligned} \tag{D.19}$$

Let σ_m be the solution of the equation $\sigma^2 = \frac{1}{\ell^2}\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)$. Note that the above equation has unique solution σ_m since σ^2 is a strictly increasing function wrt σ for $\sigma > 0$, while $\frac{1}{\ell^2}\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)$ is a strictly decreasing function wrt σ for $\sigma > 0$. If the noise has $\sigma \geq \sigma_m$, the RHS of the above equation ≥ 0, which means dice loss converges better than the regression loss.

Case 2 $\sigma \geq 1$**:** Given Tab. 4.1, if $\sigma \geq 1$, the minimum deviation in converged regression model comes from the $\mathcal{L}_1$ loss. The difference in the regression and dice loss estimates:

$$\begin{aligned}\mathbb{E}\left(\|^{r}\mathbf{w}_{\infty}-\mathbf{w}_{*}\|_{2}^{2}\right)-\mathbb{E}\left(\left\|^{d}\mathbf{w}_{\infty}-\mathbf{w}_{*}\right\|_{2}^{2}\right)\\ \propto 1-\frac{1}{\ell^{2}}\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)\end{aligned} \tag{D.20}$$

If the noise has $\sigma \geq \frac{\sqrt{2}}{\ell}\mathrm{Erf}^{-1}(\ell^2)$, the RHS of the above equation ≥ 0, which means dice loss is better than the regression loss. For objects such as cars and trailers which have length $\ell > 4m$, this is trivially satisfied.

Combining both cases, dice loss outperforms the $\mathcal{L}_1$ and $\mathcal{L}_2$ losses if the noise deviation σ exceeds the critical threshold σ_c, *i.e.*

$$\sigma > \sigma_c = \max\left(\sigma_m, \frac{\sqrt{2}}{\ell}\mathrm{Erf}^{-1}(\ell^2)\right). \tag{D.21}$$

D.1.4 Proof of Convergence Analysis Th. 2

Proof. Continuing from Lemma 3, the advantage of the trained weight obtained from dice loss over the trained weight obtained from regression losses further results in

$$\begin{aligned}
&\mathrm{Var}({}^{d}\mathbf{w}_{\infty}) \leq \mathrm{Var}({}^{r}\mathbf{w}_{\infty}) \\
\implies &\mathbb{E}(|{}^{d}\mathbf{w}_{\infty}\mathbf{h} - z|) \leq \mathbb{E}(|{}^{r}\mathbf{w}_{\infty}\mathbf{h} - z|) \\
\implies &\mathbb{E}(|\,{}^{d}\hat{z} - z|) \leq \mathbb{E}(|\,{}^{r}\hat{z} - z|) \\
\implies &\mathbb{E}({}^{d}\mathrm{IoU}_{3\mathrm{D}}) \geq \mathbb{E}({}^{r}\mathrm{IoU}_{3\mathrm{D}}),
\end{aligned} \tag{D.22}$$

assuming depth is the only source of error. Because $\mathrm{AP}_{3\mathrm{D}}$ is an non-decreasing function of $\mathrm{IoU}_{3\mathrm{D}}$, the inequality remains preserved. Hence, we have ${}^{d}\mathrm{AP}_{3\mathrm{D}} \geq {}^{r}\mathrm{AP}_{3\mathrm{D}}$. □

Thus, the average precision from the dice model is better than the regression model, which means a better detector.

D.1.5 Properties of Dice Loss.

We next explore the properties of model in Lemma 3 trained with dice loss. From Lemma 1, we write

$$\mathbb{E}\left(\left\|{}^{d}\mathbf{w}_{\infty} - \mathbf{w}_{*}\right\|_2^2\right) = c_1\mathrm{Var}(\epsilon) + c_2$$

Substituting the result of Lemma 2, we have

$$\mathbb{E}\left(\left\|{}^{d}\mathbf{w}_{\infty} - \mathbf{w}_{*}\right\|_2^2\right) = \frac{c_1}{\ell^2}\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) + c_2 \tag{D.23}$$

chapter [9] says that for a normal random variable X with mean 0 and variance 1 and for any $x > 0$, we have

$$\begin{aligned}
&\frac{\sqrt{4+x^2}-x}{2}\sqrt{\frac{1}{2\pi}}e^{-\frac{x^2}{2}} \leq P\left(X > x\right) \\
\implies &\frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq P\left(X > x\right) \\
\implies &\frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq 1 - P\left(X \leq x\right)
\end{aligned}$$

Table D.1 **Assumption comparison** of Convergence Analysis of Th. 2 vs Mono3D models.

	Th. 2	Mono3D Models
Regression	Linear	Non-linear
Noise η PDF	Normal	Arbitrary
Noise & Image	Independent	Dependent
Object Categories	1	Multiple
Object Size ℓ	Ideal	Non-ideal
Error	Depth	All 7 parameters
Loss $\mathcal{L}$	$\mathcal{L}_1$, $\mathcal{L}_2$, dice	Smooth $\mathcal{L}_1$, $\mathcal{L}_2$, dice, CE
Optimizers	SGD	SGD, Adam, AdamW
Global Optima	Unique	Multiple

$$
\begin{aligned}
\implies & \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq 1-\frac{1}{2}-\int_0^x \frac{1}{\sqrt{2\pi}}e^{-\frac{X^2}{2}}dX \\
\implies & \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq \frac{1}{2}-\int_0^x \frac{1}{\sqrt{2\pi}}e^{-\frac{X^2}{2}}dX \\
\implies & \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq \frac{1}{2}-\int_0^{\frac{x}{\sqrt{2}}} \frac{1}{\sqrt{\pi}}e^{-X^2}dX \\
\implies & \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq \frac{1}{2}-\frac{1}{2}\mathrm{Erf}\left(\frac{x}{\sqrt{2}}\right) \\
& \implies \mathrm{Erf}\left(\frac{x}{\sqrt{2}}\right) \leq 1-\frac{2}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}
\end{aligned}
$$

Substituting $x = \dfrac{\ell}{\sigma}$ above, we have,

$$
\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \leq 1-\frac{2\sigma}{\ell+\sqrt{4\sigma^2+\ell^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{\ell^2}{2\sigma^2}} \tag{D.24}
$$

Case 1: Upper bound. The RHS of Eq. (D.24) is clearly less than 1 since the term in the RHS after subtraction is positive. Hence,

$$
\mathrm{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \leq 1
$$

Substituting above in Eq. (D.23), we have

$$
\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \leq \frac{c_1}{\ell^2} + c_2 \tag{D.25}
$$

Clearly, the deviation of the trained model with the dice loss is inversely proportional to the object length ℓ. The deviation from the optimal is less for large objects.

Case 2: Infinite Noise variance $\sigma^2 \to \infty$. Then, one of the terms in the RHS of Eq. (D.24) $\frac{2\sigma}{\ell + \sqrt{4\sigma^2 + \ell^2}} \to 1$. Moreover, $\frac{\ell}{\sigma} \to 0 \implies e^{-\frac{\ell^2}{2\sigma^2}} \approx \left(1 - \frac{\ell^2}{2\sigma^2}\right)$. So, RHS of Eq. (D.24) becomes

$$\begin{aligned} \text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) &\approx 1 - \sqrt{\frac{2}{\pi}}\left(1 - \frac{\ell^2}{2\sigma^2}\right) \\ \implies \text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) &\approx \left(1 + \sqrt{\frac{2}{\pi}} + \sqrt{\frac{2}{\pi}}\frac{\ell^2}{2\sigma^2}\right) \end{aligned} \tag{D.26}$$

Substituting above in Eq. (D.23), we have

$$\begin{aligned} \mathbb{E}\left(\left\|^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \approx \frac{c_1}{\ell^2}\left(1 + \sqrt{\frac{2}{\pi}} + \sqrt{\frac{2}{\pi}}\frac{\ell^2}{2\sigma^2}\right) \\ + c_2 \end{aligned} \tag{D.27}$$

Thus, the deviation from the optimal weight is inversely proportional to the noise deviation σ^2. Hence, the deviation from the optimal weight decreases as σ^2 increases for the dice loss. This property provides noise-robustness to the model trained with the dice loss.

D.1.6 Notes on Theoretical Result

Assumption Comparisons. The theoretical result of Th. 2 relies upon several assumptions. We present a comparison between the assumptions made by Th. 2 and those underlying Mono3D models, in Tab. D.1. While our analysis depends on these assumptions, it is noteworthy that the results are apparent even in scenarios where the assumptions do not hold true. Another advantage of having a linear regression setup is that this setup has a unique global minima (because of its convexity).

Nature of Noise η. Th. 2 assumes that the noise η is a normal random variable $\mathcal{N}(0, \sigma^2)$. To verify this assumption, we take the two SoTA released models GUP Net [159] and DEVIANT [108] on the KITTI [67] Val cars. We next plot the depth error histogram of both these models in Fig. D.1. This figure confirms that the depth error is close to the Gaussian random variable. Thus, this assumption is quite realistic.

Th. 2 Requires Assumptions? We agree that Th. 2 requires assumptions for the proof. However, our theory does have empirical support; most Mono3D works have no theory. So, our theoretical

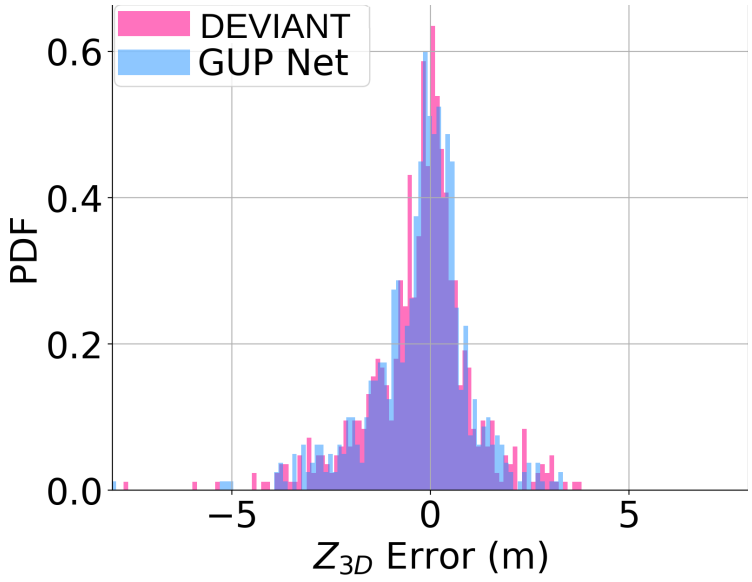


Figure D.1 **Depth error histogram** of released GUP Net and DEVIANT [108] on the KITTI Val cars. The histogram shows that depth error is close to the Gaussian random variable.

attempt for Mono3D is a step forward! We leave the analysis after relaxing some or all of these assumptions for future avenues.

Does Th. 2 Hold in Inference? Yes, Th. 2 holds even in inference. Th. 2 relies on the converged weight ${}^{\mathcal{L}}\mathbf{w}_{\infty}$, which in turn depends on the training data distribution. Now, as long as the training and testing data distribution remains the same (a fundamental assumption in ML), Th. 2 holds also during inference.

D.1.7 More Discussions

SeaBird improves because it removes depth estimation and integrates BEV segmentation. We clarify to remove this confusion. First, SeaBird also estimates depth. SeaBird depth estimates are better because of good segmentation, a *form* of depth (thanks to dice loss). Second, predicted BEV segmentation needs processing with the 3D head to output depth; so it can not replace depth estimation. Third, integrating segmentation over all categories degrades Mono3D performance ([132] and our Tab. 4.5 Sem. Category).

Why evaluation on outdoor datasets? We experiment with outdoor datasets in this chapter because indoor datasets rarely have large objects (mean length $> 6m$).

D.2 Implementation Details

Datasets. Our experiments use the publicly available KITTI-360, KITTI-360 PanopticBEV and nuScenes datasets. KITTI-360 is available at https://www.cvlibs.net/datasets/kitti-360/download.php under CCA-NonCommercial-ShareAlike (CC BY-NC-SA) 3.0 License. KITTI-360 PanopticBEV is available at http://panoptic-bev.cs.uni-freiburg.de/ under Robot Learning License Agreement. nuScenes is available at https://www.nuscenes.org/nuscenes under CC BY-NC-SA 4.0 International Public License.

Data Splits. We detail out the detection data split construction of the KITTI-360 dataset.

- *KITTI-360 Test split*: This detection benchmark [136] contains 300 training and 42 testing windows. These windows contain 61,056 training and 9,935 testing images. The calibration exists for each frame in training, while it exists for every 10^{th} frame in testing. Therefore, our split consists of 61,056 training images, while we run monocular detectors on 910 test images (ignoring uncalibrated images).
- *KITTI-360 Val split*: The KITTI-360 detection Val split partitions the official train into 239 train and 61 validation windows [136]. The original Val split [136] contains 49,003 training and 14,600 validation images. However, this original Val split has the following three issues:
  - Data leakage (common images) exists in the training and validation windows.
  - Every KITTI-360 image does not have the corresponding BEV semantic segmentation GT in the KITTI-360 PanopticBEV [72] dataset, making it harder to compare Mono3D and BEV segmentation performance.
  - The KITTI-360 validation set has higher sampling rate compared to the testing set.

  To fix the data leakage issue, we remove the common images from training set and keep them only in the validation set. Then, we take the intersection of KITTI-360 and KITTI-360 PanopticBEV datasets to ensure that every image has corresponding BEV segmentation segmentation GT. After these two steps, the training and validation set contain 48,648 and 12,408 images with calibration and semantic maps. Next, we subsample the validation images by a factor of 10 as in the testing set. Hence, our KITTI-360 Val split contains 48,648 training images and 1,294

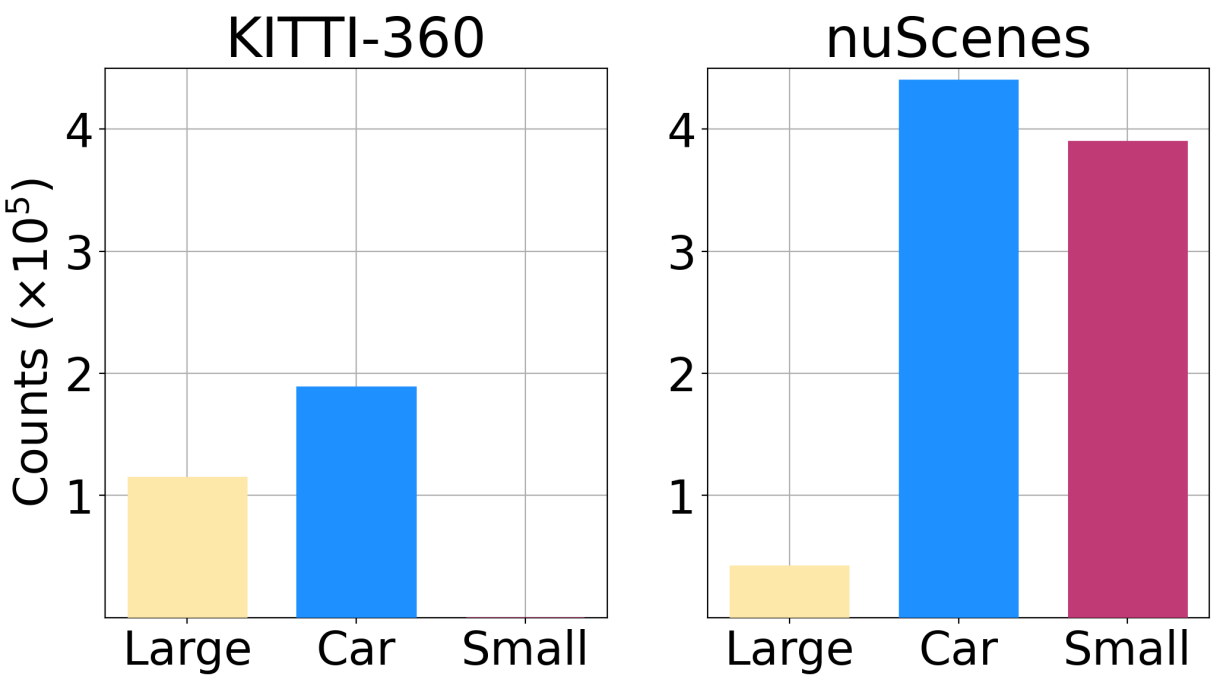


Figure D.2 **Skewness in datasets.** The ratio of large (yellow) objects to other objects is approximately 1 : 2 in KITTI-360 [136], while the skewness is about 1 : 21 in nuScenes [22].

validation images.

Augmentation. We keep the same augmentation strategy as our baselines for the respective models.

Pre-processing. We resize images to preserve their aspect ratio.

- *KITTI-360.* We resize the $[376, 1408]$ sized KITTI-360 images, and bring them to the $[384, 1438]$ resolution.
- *nuScenes.* We resize the $[900, 1600]$ sized nuScenes images, and bring them to the $[256, 704]$, $[512, 1408]$ and $[640, 1600]$ resolutions as our baselines [303, 311].

Libraries. I2M and PBEV experiments use PyTorch [184], while BEVerse and HoP use MMDetection3D [44].

Architecture.

- *I2M+SeaBird.*I2M [211] uses ResNet-18 as the backbone with the standard Feature Pyramid Network (FPN) [138] and a transformer to predict depth distribution. FPN is a bottom-up feed-forward CNN that computes feature maps with a downscaling factor of 2, and a top-down network that brings them back to the high-resolution ones. There are total four feature maps

levels in this FPN. We use the Box Net with ResNet-18 [77] as the detection head.

- *PBEV+SeaBird.*PBEV [72] uses EfficientDet [231] as the backbone. We use Box Net with ResNet-18 [77] as the detection head.
- *BEVerse+SeaBird.* BEVerse [303] uses Swin transformers [152] as the backbones. We use the original heads without any configuration change.
- *HoP+SeaBird.* HoP [311] uses ResNet-50, ResNet-101 [77] and V2-99 [181] as the backbones. Since HoP does not have the segmentation head, we use the one in BEVerse as the segmentation head.

We initialize the CNNs and transformers from ImageNet weights except for V2-99, which is pre-trained on 15 million LiDAR data.. We output two and ten foreground categories for KITTI-360 and nuScenes datasets respectively.

Training. We use the training protocol as our baselines for all our experiments. We choose the model saved in the last epoch as our final model for all our experiments.

- *I2M+SeaBird.* Training uses the Adam optimizer [102], a batch size of 30, an exponential decay of 0.98 [211] and gradient clipping of 10 on single Nvidia A100 (80GB) GPU. We train the BEV Net in the first stage with a learning rate 1.0×10^{-4} for 50 epochs [211] . We then add the detector in the second stage and finetune with the first stage weight with a learning rate 0.5×10^{-4} for 40 epochs. Training on KITTI-360 Val takes a total of 100 hours. For Test models, we finetune I2M Val stage 1 model with train+val data for 40 epochs.
- *PBEV+SeaBird.* Training uses the Adam optimizer [102] with Nesterov, a batch size of 2 per GPU on eight Nvidia RTX A6000 (48GB) GPU. We train the PBEV with the dice loss in the first stage with a learning rate 2.5×10^{-3} for 20 epochs. We then add the Box Net in the second stage and finetune with the first stage weight with a learning rate 2.5×10^{-3} for 20 epochs. PBEV decays the learning rate by 0.5 and 0.2 at 10 and 15 epoch respectively. Training on KITTI-360 Val takes a total of 80 hours. For Test models, we finetune PBEV Val stage 1 model with train+val data for 10 epochs on four GPUs.
- *BEVerse+SeaBird.* Training uses the AdamW optimizer [156], a sample size of 4 per GPU,

the one-cycle policy [303] and gradient clipping of 35 on eight Nvidia RTX A6000 (48GB) GPU [303]. We train the segmentation head in the first stage with a learning rate 2.0×10^{-3} for 4 epochs. We then add the detector in the second stage and finetune with the first stage weight with a learning rate 2.0×10^{-3} for 20 epochs [303]. Training on nuScenes takes a total of 400 hours.

- *HoP+SeaBird.* Training uses the AdamW optimizer [156], a sample size of 2 per GPU, and gradient clipping of 35 on eight Nvidia A100 (80GB) GPUs [311]. We train the segmentation head in the first stage with a learning rate 1.0×10^{-4} for 4 epochs. We then add the detector in the second stage and finetune with the first stage weight with a learning rate 1.0×10^{-4} for 24 epochs [303]. nuScenes training takes a total of 180 hours. For Test models, we finetune val model with train+val data for 4 more epochs.

Losses. We train the BEV Net of SeaBird in Stage 1 with the dice loss. We train the final SeaBird pipeline in Stage 2 with the following loss:

$$\mathcal{L} = \mathcal{L}_{det} + \lambda_{seg}\mathcal{L}_{seg}, \tag{D.28}$$

with $\mathcal{L}_{seg}$ being the dice loss and λ_{seg} being the weight of the dice loss in the baseline. We keep the $\lambda_{seg} = 5$. If the segmentation loss is itself scaled such as PBEV uses the $\mathcal{L}_{seg}$ as 7, we use $\lambda_{seg} = 35$ with detection.

Inference. We report the performance of all KITTI-360 and nuScenes models by inferring on single GPU card. Our testing resolution is same as the training resolution. We do not use any augmentation for test/validation.

We keep the maximum number of objects is 50 per image for KITTI-360 models. We use score threshold of 0.1 for KITTI-360 models and class dependent threshold for nuScenes models as in [303]. KITTI-360 evaluates on windows and not on images. So, we use a 3D center-based NMS [109] to convert image-based predictions to window-based predictions for SeaBird and all our KITTI-360 baselines. This NMS uses a threshold of 4m for all categories, and keeps the highest score 3D box if multiple 3D boxes exist inside a window.

Table D.2 **Error analysis** on KITTI-360 Val.

Oracle							AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)		
x	y	z	l	w	h	θ	AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP
							8.71	43.19	25.95	35.76	52.22	43.99
✓							9.78	41.63	25.70	36.07	50.63	43.35
	✓						9.57	46.08	27.82	34.65	53.03	43.84
		✓					9.90	42.32	27.11	39.66	53.08	46.37
✓	✓	✓					19.90	47.37	33.63	41.84	52.53	47.19
			✓	✓	✓		9.49	45.67	27.58	33.43	51.53	42.48
✓	✓	✓	✓	✓	✓		37.09	46.27	41.68	44.58	51.15	47.87
✓	✓	✓	✓	✓	✓	✓	37.02	47.03	42.02	44.46	51.50	47.98

Table D.3 **Complexity analysis** on KITTI-360 Val.

Method	Mono3D	Inf. Time (s)	Param (M)	Flops (G)
GUP Net [159]	✓	0.02	16	30
DEVIANT [108]	✓	0.04	16	235
I2M [211]	×	0.01	40	80
I2M+SeaBird	✓	0.02	53	130
PBEV [72]	×	0.14	24	229
PBEV+SeaBird	✓	0.15	37	279

D.3 Additional Experiments and Results

We now provide additional details and results of the experiments evaluating SeaBird's performance.

D.3.1 KITTI-360 Val Results

Error Analysis. We next report the error analysis of the SeaBird in Tab. D.2 by replacing the predicted box data with the oracle box data as in [166]. We consider the GT box to be an oracle box for predicted box if the euclidean distance is less than $4m$. In case of multiple GT being matched to one box, we consider the oracle with the minimum distance. Tab. D.2 shows that depth is the biggest source of error for Mono3D task as also observed in [166]. Moreover, the oracle does not lead to perfect results since the KITTI-360 PanopticBEV GT BEV semantic is only upto $50m$, while the KITTI-360 evaluates all objects (including objects beyond $50m$).

Computational Complexity Analysis. We next compare the complexity analysis of SeaBird pipeline in Tab. D.3. For the flops analysis, we use the fvcore library as in [108].

Naive baseline for Large Objects. We next compare SeaBird against a naive baseline for large objects detection, such as by fine-tuning GUP Net only on larger objects. Tab. D.4 shows that

Table D.4 **KITTI-360 Val results with naive baseline finetuned for large objects.** SeaBird pipelines comfortably outperform this naive baseline on large objects. [Key: **Best**, **Second Best**, †= Retrained]

Method	Venue	AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)			BEV Seg IoU [%](↑)		
		AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
GUP Net† [159]	ICCV21	0.54	**45.11**	22.83	0.98	50.52	25.75	–	–	–
GUP Net (Large FT)† [159]	ICCV21	0.56	–	0.28	2.56	–	1.28	–	–	–
I2M+SeaBird	CVPR24	**8.71**	**43.19**	**25.95**	**35.76**	**52.22**	**43.99**	**23.23**	**39.61**	**31.42**
PBEV+SeaBird	CVPR24	**13.22**	42.46	**27.84**	**37.15**	**52.53**	**44.84**	**24.30**	**48.04**	**36.17**

Table D.5 **Impact of denoising** BEV segmentation maps with MIRNet-v2 [294] on KITTI-360 Val with I2M+SeaBird. Denoising does not help. [Key: **Best**]

Denoiser	AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)			BEV Seg IoU [%](↑)		
	AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
✓	2.73	**43.77**	23.25	14.34	51.23	32.79	21.42	**39.72**	30.57
×	**8.71**	43.19	**25.95**	**35.76**	**52.22**	**43.99**	**23.23**	39.61	**31.42**

Table D.6 **Segmentation loss weight λ_{seg} sensitivity** on KITTI-360 Val with I2M+SeaBird. $\lambda_{seg} = 5$ works the best. [Key: **Best**]

λ_{seg}	AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)			BEV Seg IoU [%](↑)		
	AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
0	4.86	**45.09**	24.98	26.33	52.31	39.32	0	7.07	3.54
1	7.07	41.71	24.39	32.92	52.9	42.91	23.78	40.58	32.18
3	7.26	43.45	25.36	34.47	**52.54**	43.51	**23.40**	**40.15**	**31.78**
5	**8.71**	43.19	**25.95**	**35.76**	52.22	**43.99**	23.23	39.61	31.42
10	7.69	43.41	25.55	34.22	50.97	42.60	22.15	39.83	30.99

SeaBird pipelines comfortably outperform this baseline as well.

Does denoising BEV images help? Another potential addition to the SeaBird framework is using a denoiser between segmentation and detection heads. We use the MIRNet-v2 [294] as our denoiser and train the BEV segmentation head, denoiser and detection head in an end-to-end manner. Tab. D.5 shows that denoising does not increase performance but the inference time. Hence, we do not use any denoiser for SeaBird.

Sensitivity to Segmentation Weight. We next study the impact of segmentation weight on I2M+SeaBird in Tab. D.6 as in Sec. 4.4.2. Tab. D.6 shows that $\lambda_{seg} = 5$ works the best for the Mono3D of large objects.

Reproducibility. We ensure reproducibility of our results by repeating our experiments for 3 random seeds. We choose the final epoch as our checkpoint in all our experiments as [108]. Tab. D.7 shows the results with these seeds. SeaBird outperforms SeaBird without dice loss in the

Table D.7 **Reproducibility results** on KITTI-360 Val with I2M+SeaBird. SeaBird outperforms SeaBird without dice loss in the median and average cases. [Key: **Best**, **Second Best**]

Dice	Seed	AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)			BEV Seg IoU [%](↑)		
		AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
×	111	3.81	44.63	24.22	24.96	53.15	39.06	0	5.99	3.00
	444	4.86	45.09	24.98	26.33	52.31	39.32	0	7.07	3.54
	222	5.79	46.71	26.25	24.32	54.06	39.19	0	5.32	2.66
	Avg	4.82	45.58	**25.15**	25.20	53.17	**39.19**	0	6.13	3.06
✓	111	7.87	44.03	25.95	33.55	53.93	43.74	22.64	40.64	31.64
	444	8.71	43.19	25.95	35.76	52.22	43.99	23.23	39.61	31.42
	222	8.71	42.87	25.79	34.71	51.72	43.22	22.74	40.01	31.38
	Avg	8.43	43.36	**25.90**	34.67	52.62	**43.65**	22.87	40.09	31.48

Table D.8 **Dice vs regression on methods with depth estimation**. Dice model again outperforms regression loss models, particularly for large objects. [Key: **Best**, **Second Best**]

Resolution	Method	BBone	Venue	Loss	AP_{Lrg} (↑)	AP_{Car} (↑)	AP_{Sml} (↑)	mAP (↑)	NDS (↑)
256×704	HoP+SeaBird	R50	ICCV23	–	**27.4**	**57.2**	**46.4**	**39.9**	**50.9**
			–	$\mathcal{L}_1$	27.0	57.1	46.5	39.7	50.7
			–	$\mathcal{L}_2$	Did Not Converge				
			CVPR24	Dice	**28.2**	**58.6**	**47.8**	**41.1**	**51.5**

median and average cases. The biggest improvement shows up on larger objects.

D.3.2 nuScenes Results

Extended Val Results. Besides showing improvements upon existing detectors in Tab. 4.7 on the nuScenes Val split, we compare with more recent SoTA detectors with large backbones in Tab. D.9.

Dice vs regression on depth estimation methods. We report HoP +R50 config, which uses depth estimation and compare losses in Tab. D.8. Tab. D.8 shows that Dice model again outperforms regression loss models.

SeaBird Compatible Approaches. SeaBird conditions the detection outputs on segmented BEV features and so, requires foreground BEV segmentation. So, all approaches which produce latent BEV map in Tabs. 4.6 and 4.7 are compatible with SeaBird. However, approaches which do not produce BEV features such as SparseBEV [142] are incompatible with SeaBird.

D.3.3 Qualitative Results

KITTI-360. We now show some qualitative results of models trained on KITTI-360 Val split in Fig. D.3. We depict the predictions of PBEV+SeaBird in image view on the left, the predictions of PBEV+SeaBird, the baseline MonoDETR [300], predicted and GT boxes in BEV in the mid-

Table D.9 **nuScenes Val Detection results**. SeaBird pipelines outperform the baselines, particularly for large objects. [Key: **Best**, **Second Best**, B= Base, S= Small, T= Tiny, ♙= Released, *= Reimplementation, §= CBGS]

Resolution	Method	BBone	Venue	AP_{Lrg} (↑)	AP_{Car} (↑)	AP_{Sml} (↑)	mAP (↑)	NDS (↑)
256×704	CAPE♙ [273]	R50	CVPR23	18.5	53.2	38.1	31.8	44.2
	PETRv2 [150]	R50	ICCV23	–	–	–	34.9	45.6
	SOLOFusion♙§ [182]	R50	ICLR23	26.5	**57.3**	**48.5**	**40.6**	49.7
	BEVerse-T♙ [303]	Swin-T	ArXiv	18.5	53.4	38.8	32.1	46.6
	BEVerse-T+SeaBird	Swin-T	CVPR24	19.5	54.2	41.1	33.8	48.1
	HoP♙ [311]	R50	ICCV23	**27.4**	57.2	46.4	39.9	**50.9**
	HoP+SeaBird	R50	CVPR24	**28.2**	**58.6**	**47.8**	**41.1**	**51.5**
512×1408	3DPPE [219]	R101	ICCV23	–	–	–	39.1	45.8
	STS [261]	R101	AAAI23	–	–	–	43.1	52.5
	P2D [101]	R101	ICCV23	–	–	–	43.3	52.8
	BEVDepth [127]	R101	AAAI23	–	–	–	41.8	53.8
	BEVDet4D [87]	R101	ArXiv	–	–	–	42.1	54.5
	BEVerse-S♙ [303]	Swin-S	ArXiv	20.9	56.2	42.2	35.2	49.5
	BEVerse-S+SeaBird	Swin-S	CVPR24	24.6	58.7	45.0	38.2	51.3
	HoP* [311]	R101	ICCV23	**31.4**	**63.7**	**52.5**	**45.2**	**55.0**
	HoP+SeaBird	R101	CVPR24	**32.9**	**65.0**	**53.1**	**46.2**	**54.7**
640×1600	BEVDet [88]	V2-99	ArXiv	29.6	61.7	48.2	42.1	48.2
	PETRv2 [150]	R101	ICCV23	–	–	–	42.1	52.4
	CAPE♙ [273]	V2-99	CVPR23	31.2	63.2	51.9	44.7	54.4
	BEVDet4D§ [87]	Swin-B	ArXiv	–	–	–	42.6	55.2
	HoP* [311]	V2-99	ICCV23	**36.5**	**69.1**	**56.1**	**49.6**	**58.3**
	HoP+SeaBird	V2-99	CVPR24	**40.3**	**71.7**	**58.8**	**52.7**	**60.2**
900×1600	FCOS3D [250]	R101	ICCVW21	–	–	–	34.4	41.5
	PGD [251]	R101	CoRL21	–	–	–	36.9	42.8
	DETR3D [257]	R101	CoRL21	22.4	60.3	41.1	34.9	43.4
	PETR [149]	R101	ECCV22	–	–	–	37.0	44.2
	BEVFormer [132]	R101	ECCV22	27.7	48.5	34.5	41.5	51.7
	PolarFormer [95]	V2-99	AAAI23	–	–	–	50.0	56.2

dle and BEV semantic segmentation predictions from PBEV+SeaBird on the right. In general, PBEV+SeaBird detects more larger objects (buildings) than GUP Net [159].

nuScenes. We now show some qualitative results of models trained on nuScenes Val split in Fig. D.4. As before, we depict the predictions of BEVerse-S+SeaBird in image view from six cameras on the left and BEV semantic segmentation predictions from SeaBird on the right.

KITTI-360 Demo Video. We next put a short demo video of PBEV+SeaBird model trained on KITTI-360 Val split compared with MonoDETR at

https://www.youtube.com/watch?v=SmuRbMbsnZA. We run our trained model independently on each frame of KITTI-360. None of the frames from the raw video appear in the training set of KITTI-360 Val split. We use the camera matrices available with the video but do not use any

temporal information. Overlaid on each frame of the raw input videos, we plot the projected 3D boxes of the predictions, predicted and GT boxes in BEV in the middle and BEV semantic segmentation predictions from PBEV+SeaBird. We set the frame rate of this demo at 5 fps similar to [108]. The demo video demonstrates impressive results on larger objects.

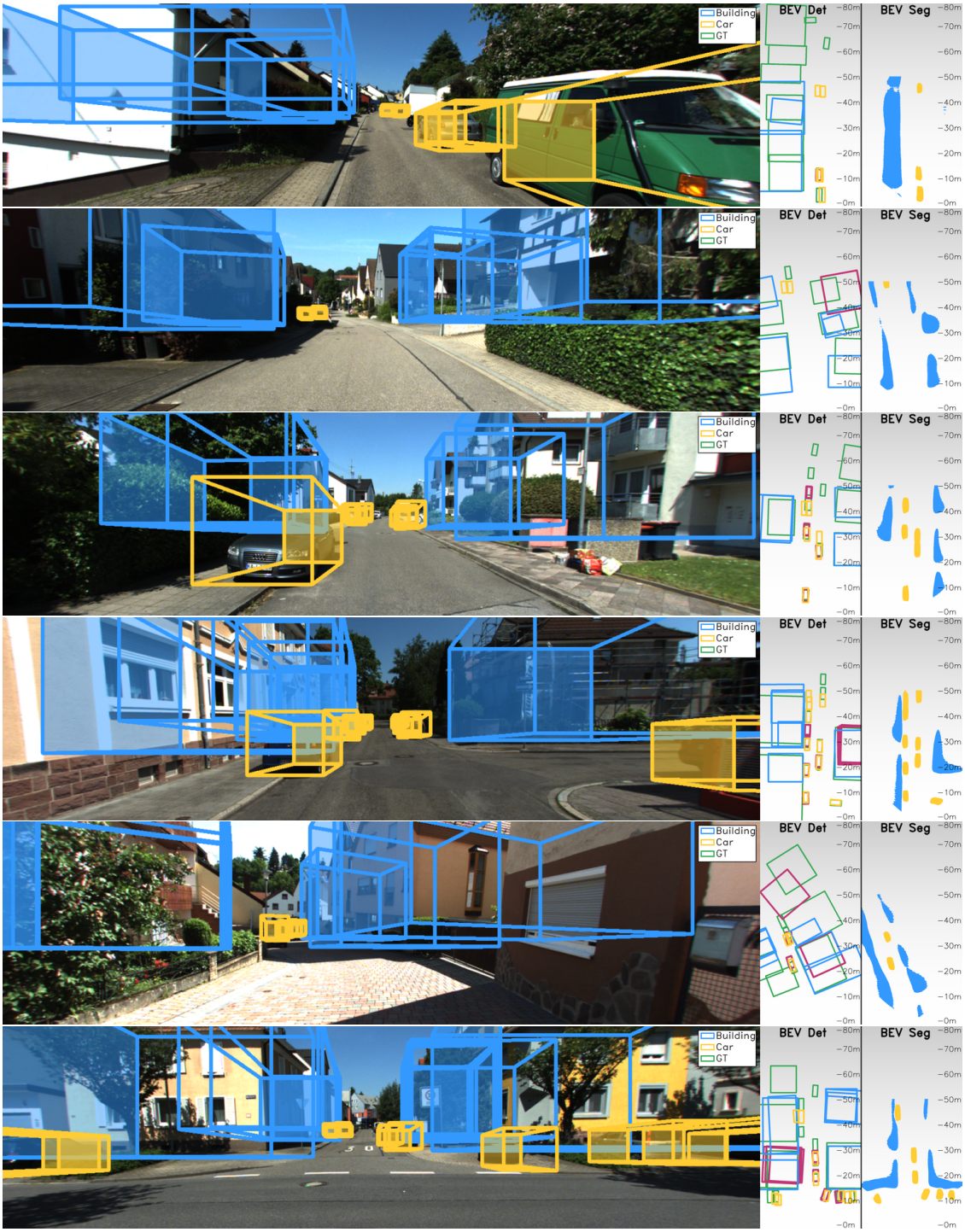


Figure D.3 **KITTI-360 Qualitative Results**. PBEV+SeaBird detects more large objects (buildings, in blue) than MonoDETR [300] in orange. We depict the predictions of PBEV+SeaBird in the image view on the left, the predictions of PBEV+SeaBird, the baseline MonoDETR [300], and ground truth in BEV in the middle, and BEV semantic segmentation predictions from PBEV+SeaBird on the right. [Key: Buildings (in blue) and Cars (in yellow) of PBEV+SeaBird; all classes (pink) of MonoDETR [300], and Ground Truth (in green) in BEV].

Figure D.4 **nuScenes Qualitative Results**. The first row shows the front_left, front, and front_right cameras, while the second row shows the back_left, back, and back_right cameras. [Key: Cars (blue), Vehicles (green), Pedestrian (violet), Cones (yellow) and Barrier (gray) of BEVerse-S+SeaBird at 200×200 resolution in BEV].

APPENDIX E

CHARM3R APPENDIX

E.1 Additional Details and Proof

We now add more details and proofs which we could not put in the main paper because of the space constraints.

E.1.1 Proof of Ground Depth Lemma 1

We reproduce the proof from [284] with our notations for the sake of completeness of this work.

Proof. We first rewrite the pinhole projection Eq. (5.1) as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}^{-1}(\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} z - \mathbf{T}). \tag{E.1}$$

We now represent the ray shooting from the camera optical center through each pixel as $\overrightarrow{r}(u, v, z)$. Using the matrix $\boldsymbol{A} = (a_{ij}) = \mathbf{R}^{-1}\mathbf{K}^{-1}$, and the vector $\boldsymbol{B} = (b_i) = -\mathbf{R}^{-1}\mathbf{T}$, we define the parametric ray as:

$$\overrightarrow{r}(u, v, z) : \begin{cases} X = (a_{11}u + a_{12}v + a_{13})z + b_1 \\ Y = (a_{21}u + a_{22}v + a_{23})z + b_2 \\ Z = (a_{31}u + a_{32}v + a_{33})z + b_3 \end{cases} \tag{E.2}$$

Moreover, the ground at a distance h can be described by a plane, which is determined by the point $(0, H, 0)$ in the plane and the normal vector $\overrightarrow{n} = (0, 1, 0)$:

$$\overrightarrow{r} \cdot \overrightarrow{n} = H. \tag{E.3}$$

Then, the ground depth is the intersection point between this ray and the ground plane. Combining Eqs. (E.2) and (E.3), the ground depth z of the pixel (u, v) is:

$$\begin{aligned} &(a_{21}u + a_{22}v + a_{23})z + b_2 = H \\ &\qquad \implies z = \frac{H - b_2}{a_{21}u + a_{22}v + a_{23}}. \end{aligned} \tag{E.4}$$

□

E.1.2 Proof of Lemma 5

We next derive Lemma 5 from Lemma 4 as follows.

Proof.

$$\boldsymbol{A} = (a_{ij}) = \mathbf{R}^{-1}\mathbf{K}^{-1} = \boldsymbol{I}^{-1} \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1}$$
$$= \boldsymbol{I} \begin{bmatrix} \frac{1}{f} & 0 & \frac{-u_0}{f} \\ 0 & \frac{1}{f} & \frac{-v_0}{f} \\ 0 & 0 & 1 \end{bmatrix},$$

with rotation matrix $\mathbf{R}$ is identity $\boldsymbol{I}$ for forward cameras. So, $a_{21} = 0, a_{22} = \frac{1}{f}, a_{23} = \frac{-v_0}{f}$. Substituting a_{21}, a_{22}, a_{23} in Eq. (5.2), we get Eq. (5.3). □

E.1.3 Extension to Camera not parallel to Ground

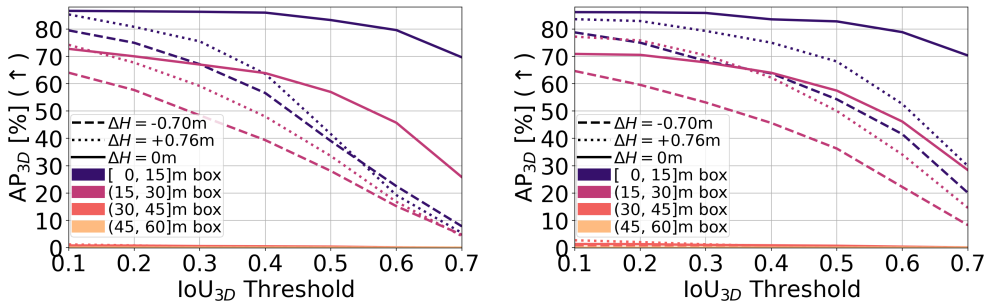
Following Sec. 3.3 of GEDepth [284], we use the camera pitch δ, and generalize Eq. (5.2) to obtain ground depth as

$$z = \frac{H - b_2\cos\delta - b_3\sin\delta}{[a_{21}u + a_{22}v + a_{23}]\cos\delta + [a_{31}u + a_{32}v + a_{33}]\sin\delta}$$
$$= \frac{H - b_2\cos\delta - b_3\sin\delta}{\frac{v - v_0}{f}\cos\delta + \sin\delta} \tag{E.5}$$

. Note that if camera pitch $\delta = 0$, this reduces to the usual form of Eq. (5.2) and Eq. (5.3) respectively. Also, Th. 1 has a more general form with the pitch value, and remains valid for majority of the pitch angle ranges.

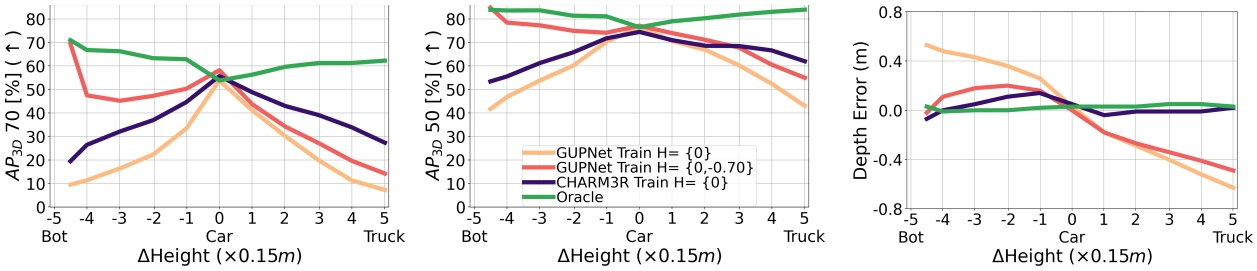
E.1.4 Extension to Not-flat Roads

For non-flat roads, we assume that the road is made of multiple flat 'pieces' of roads each with its own slope and we predict the slope of each pixel as in GEDepth [284]. To predict slope $\hat{\delta}$ of each pixel, we first define a set of N discrete slopes: $\{\tau_i, i = 1, ..., N\}$. We compute each pixel slope by linearly combining the discrete slopes with the predicted probability distribution



(a) GUP Net (b) CHARM3R

Figure E.1 **CARLA Val AP_{3D} at different depths and IoU_{3D} thresholds** with GUP Net. CHARM3R shows biggest gains on $IoU_{3D} > 0.3$ for $[0, 30]m$ boxes.



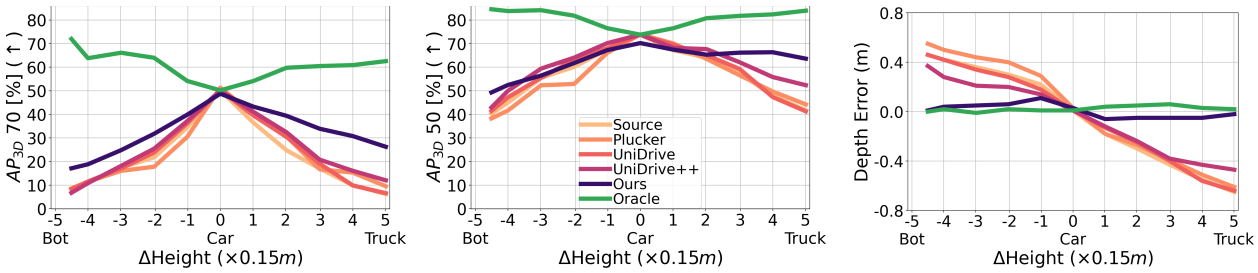
(a) $AP_{3D}\,70$ [%] comparison. (b) $AP_{3D}\,50$ [%] comparison. (c) MDE comparison.

Figure E.2 **CARLA Val Results with GUP Net** detector after augmentation of [104]. Training a detector with both $\Delta H = -0.70m$ and $\Delta H = 0m$ images produces better results at $\Delta H = -0.70m$ and $\Delta H = 0m$, but **fails at unseen height images** $\Delta H = +0.76m$. CHARM3R **outperforms** all baselines, especially at unseen bigger height changes. All methods except Oracle are trained on car height and tested on all heights.

$\{\hat{p}_i \in [0,1], \sum_i \hat{p}_i = 1\}$ over N slopes $\hat{\delta} = \sum_i \hat{p}_i \tau_i$. We train the network to minimize the total loss: $L_{\text{total}} = L_{\text{det}} + \lambda_{\text{slope}} L_{\text{slope}}(\delta, \hat{\delta})$, where L_{det} are the detection losses, and L_{slope} is the slope classification loss. We next substitute the predicted slope in [**?**]. We do not run this experiment since planar ground is reasonable assumption for most driving scenarios within some distance.

E.1.5 Unrealistic assumptions of Th. 4

We partially agree. These assumptions reflect the observations of [51]. Also, our theory has empirical support; most Mono3D works have no theory. So, our theoretical attempt is a step forward!



(a) AP_{3D} 70 [%] comparison. (b) AP_{3D} 50 [%] comparison. (c) MDE comparison.

Figure E.3 **CARLA Val Results with DEVIANT** detector. CHARM3R **outperforms** all baselines, especially at bigger height changes. All methods except Oracle are trained on car height and tested on all heights. Results of inference on height changes of $-0.70, 0$ and 0.76 meters are in Tab. 5.2.

E.2 Additional Experiments

We now provide additional details and results of the experiments evaluating CHARM3R's performance.

E.2.1 CARLA Val Results

We first analyze the results on the synthetic CARLA dataset further.

AP at different distances and thresholds. We next compare the AP_{3D} of the baseline GUP Net and CHARM3R in Fig. E.1 at different distances in meters and IoU_{3D} matching criteria of $0.1-0.7$ as in [108]. Fig. E.1 shows that CHARM3R is effective over GUP Net at all depths and higher IoU_{3D} thresholds. CHARM3R shows biggest gains on $IoU_{3D} > 0.3$ for $[0, 30]m$ boxes.

Comparison with Augmentation-Methods. Sec. 5.1 of the paper says that the augmentation strategy falls short when the target height is OOD. We show this in Fig. E.2. Since authors of [104] do not release the NVS code, we use the ground truth images from height change $\Delta H = -0.70m$ in training. Fig. E.2 confirms that augmentation also improves the performance on $\Delta H = -0.70m$ and $\Delta H = 0m$, but again falls short on unseen ego heights $\Delta H = +0.76m$. On the other hand, CHARM3R (even though trained on $\Delta H = -0.70m$) outperforms such augmentation strategy at unseen ego heights $\Delta H = +0.76m$. This shows the complementary nature of CHARM3R over augmentation strategies.

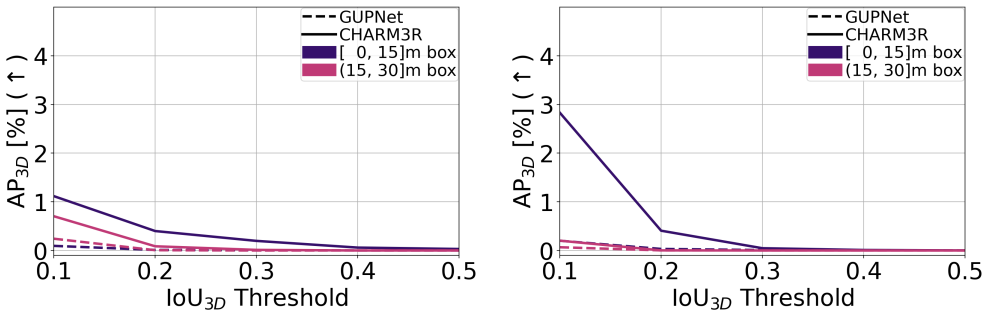
Reproducibility. We ensure reproducibility of our results by repeating our experiments for 3 random seeds. We choose the final epoch as our checkpoint in all our experiments as [108, 110]. Tab. E.1 shows the results with these seeds. CHARM3R outperforms the baseline GUP Net in both

Table E.1 **Reproducibility Results.** CHARM3R **outperforms** all other baselines on CARLA Val split, especially at bigger unseen ego heights in both median (Seed=444) and average cases. All except Oracle are trained on car height $\Delta H = 0m$ and tested on bot to truck height data. [Key: **Best**]

3D Detector	Seed ↓ / ΔH (m)→	AP_{3D} 70 [%] (↑)			AP_{3D} 50 [%] (↑)			MDE (m) [≈ 0]		
		−0.70	0	+0.76	−0.70	0	+0.76	−0.70	0	+0.76
GUP Net [159]	111	12.24	55.98	7.53	44.14	76.37	41.32	+0.48	+0.00	−0.64
	444	9.46	53.82	7.23	41.66	76.47	40.97	+0.53	+0.03	−0.63
	222	10.35	52.94	10.79	41.67	75.80	46.45	+0.53	+0.01	−0.57
	Average	10.68	54.25	8.52	42.49	**76.21**	43.58	+0.51	+0.01	−0.61
+ CHARM3R	111	19.99	58.16	29.96	54.15	74.10	64.27	+0.09	+0.00	−0.03
	444	19.45	55.68	27.33	53.40	74.47	61.98	+0.07	+0.05	−0.02
	222	17.41	53.57	27.77	54.30	74.83	64.42	+0.12	+0.01	−0.09
	Average	**18.95**	**55.80**	**28.35**	**53.95**	74.47	**63.56**	+0.09	+0.02	−0.05
Oracle	–	70.96	53.82	62.25	83.88	76.47	83.96	+0.03	+0.03	+0.03

Table E.2 **nuScenes to CODa Val Results.** CHARM3R **outperforms** all baselines, especially at unseen height changes. [Key: **Best**, **Second Best**, Ped= Pedestrians]

3D Detector	Method	Car AP_{3D} 50 [%] (↑)		Ped AP_{3D} 30 [%] (↑)	
		CODa	nuScenes	CODa	nuScenes
GUP Net [159]	Source	0.02	**18.42**	0.01	**2.93**
	UniDrive [129]	0.02	**18.42**	0.01	**2.93**
	UniDrive++[129]	**0.03**	**18.42**	**0.02**	**2.93**
	CHARM3R	**0.30**	**14.80**	**0.05**	**1.26**
	Oracle	28.56	18.42	30.31	2.93



(a) CODa Car (b) CODa Pedestrian

Figure E.4 **CODa Val AP_{3D} at different depths and IoU_{3D} thresholds** with GUP Net trained on nuScenes. CHARM3R shows biggest gains on $IoU_{3D} < 0.3$ for $[0, 30]m$ boxes.

median and average cases.

Results with DEVIANT. We next additionally plot the robustness of CHARM3R with other methods on the DEVIANT detector [108] in Fig. E.3 The figure confirms that CHARM3R works even with DEVIANT and produces SoTA robustness to unseen ego heights.

E.2.2 nuScenes ⇀ CODa Val Results

To test our claims further in real-life, we use two real datasets: the nuScenes dataset [22] and the recently released CODa [295] datasets. nuScenes has ego camera at height $1.51m$ above the ground, while the CODa is a robotics dataset with ego camera at a height of $0.75m$ above the ground. This experiment uses the following data split:

- *nuScenes Val Split.* This split [22] contains 28,130 training and 6,019 validation images from the front camera as [108].
- *CODa Val Split.* This split [295] contains 19,511 training and 4,176 validation images. We only use this split for testing.

We train the GUP Net detector with 10 nuScenes classes and report the results with the KITTI metrics on both nuScenes val and CODa Val splits.

Main Results. We report the main results in Tab. E.2 paper. The results of Tab. E.2 shows gains on both Cars and Pedestrians classes of CODa val dataset. The performance is very low, which we believe is because of the domain gap between nuScenes and CODa datasets. These results further confirm our observations that unlike 2D detection, generalization across unseen datasets remains a big problem in the Mono3D task.

AP at different distances and thresholds. To further analyze the performance, we next plot the AP_{3D} of the baseline GUP Net and CHARM3R in Fig. E.4 at different distances in meters and IoU_{3D} matching criteria of $0.1 - 0.5$ as in [108]. Fig. E.4 shows that CHARM3R is effective over GUP Net at all depths and lower IoU_{3D} thresholds. CHARM3R shows biggest gains on $IoU_{3D} < 0.3$ for $[0, 30]m$ boxes. The gains are more on the Pedestrian class on CODa since CODa captures UT Austin campus scenes, and therefore, has more pedestrians compared to cars. nuScenes captures outdoor driving scenes in Boston and Singapore, and therefore, has more cars compared to pedestrians. We describe the statistics of these two datasets in Tab. E.3.

E.2.3 Qualitative Results.

CARLA. We now show some qualitative results of models trained on CARLA Val split from car height ($\Delta H = 0m$) and tested on truck height ($\Delta H = +0.76m$) in Fig. E.5. We depict the

Table E.3 **Dataset statistics.** nuScenes Val has more Cars compared to Pedestrians, while CODa Val has more Pedestrians than Cars.

Val	**Ego Ht** (m)	**#Images**	**Car** (k)	**Ped** (k)
nuScenes	1.51	6,019	18	7
CODa	0.75	4,176	4	86

predictions of CHARM3R in image view on the left, the predictions of CHARM3R, the baseline GUP Net [159], and GT boxes in BEV on the right. In general, CHARM3R detects objects more accurately than GUP Net [159], making CHARM3R more robust to camera height changes. The regression-based baseline GUP Net mostly underestimates the depth of 3D boxes with positive ego height changes, which qualitatively justifies the claims of Th. 4.

CODa. We now show some qualitative results of models trained on CODa Val split in Fig. E.6. As before, we depict the predictions of CHARM3R in image view image view on the left, the predictions of CHARM3R, the baseline GUP Net [159], and GT boxes in BEV on the right. In general, CHARM3R detects objects more accurately than the baseline GUP Net [159], making CHARM3R more robust to camera height changes. Also, considerably less number of boxes are detected in the cross-dataset evaluation *i.e.* on CODa Val. We believe this happens because of the domain shift.

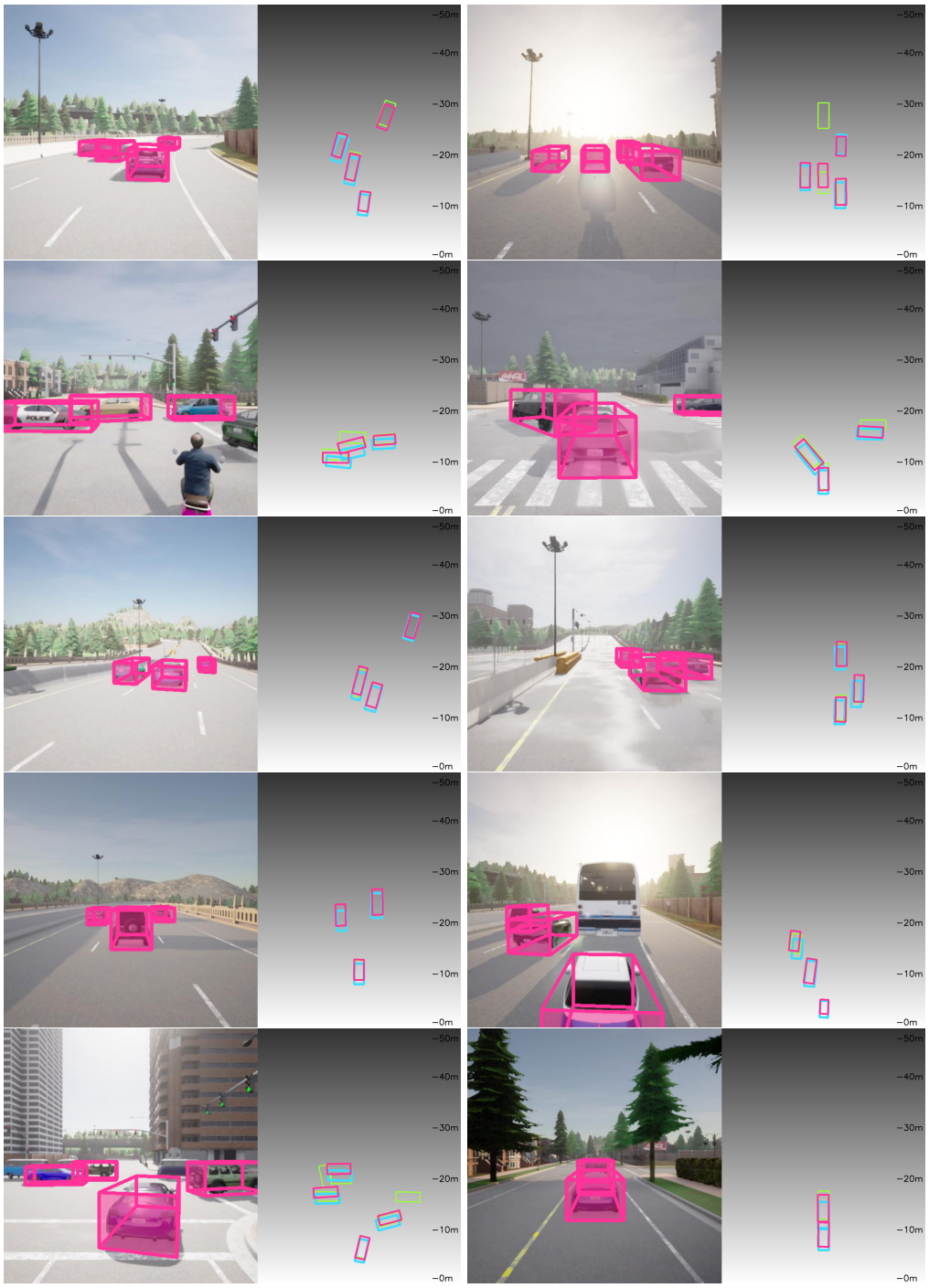


Figure E.5 **CARLA Val Qualitative Results**. CHARM3R detects objects more accurately than GUP Net [159], making CHARM3R more robust to camera height changes. The regression-based baseline GUP Net mostly underestimates the depth which qualitatively justifies the claims of Th. 4. All methods are trained on CARLA images at car height $\Delta H = 0m$ and evaluated on $\Delta H = +0.76m$. [Key: Cars (pink) of CHARM3R. ; Cars (cyan) of GUP Net, and Ground Truth (green) in BEV.

Figure E.6 **CODa Val Qualitative Results**. CHARM3R detects objects more accurately than GUP Net [159], making CHARM3R more robust to camera height changes. All methods are trained on nuScenes dataset and evaluated on CODa dataset. [Key: Cars (pink) and Pedestrian (violet) of CHARM3R. ; all classes (cyan) of GUP Net, and Ground Truth (green) in BEV.